The Pennsylvania State University
The Graduate School

LEARNING IN EXTREME CONDITIONS: ONLINE AND ACTIVE LEARNING WITH MASSIVE, IMBALANCED AND NOISY DATA

A Dissertation in
Computer Science and Engineering
by
Şeyda Ertekin

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2009

The dissertation of Şeyda Ertekin was reviewed and approved* by the following:

C. Lee Giles
David Reese Professor of College of Information Sciences and Technology
Dissertation Advisor, Chair of Committee

Léon Bottou
Distinguished Research Scientist
NEC Laboratories America Inc., Princeton, NJ 08540
Special Member

Raj Acharya
Professor of Computer Science and Engineering
Head of the Computer Science and Engineering Department

Jia Li
Associate Professor of Statistics Department

Tracy Mullen
Assistant Professor of College of Information Sciences and Technology

*Signatures are on file in the Graduate School.

Abstract

This thesis addresses improving the performance of machine learning algorithms with a particular focus on classification tasks with large, imbalanced and noisy datasets. The field of machine learning addresses the question of how best to use experimental or historical data to discover general patterns and regularities and improve the process of decision making. However, applying machine learning algorithms to very large scale problems still poses challenges. Additionally, class imbalance and noise in the data degrade the prediction accuracy of standard machine learning algorithms. The main focus of this thesis is designing machine learning algorithms and approaches that are faster, data efficient and less demanding in computational resources to achieve scalable algorithms for large scale problems. This thesis addresses these problems in active and online learning frameworks. The particular focus of the thesis is on Support Vector Machine (SVM) algorithm with classification problems, but the proposed approaches on active and online learning are also well extensible to other widely used machine learning algorithms.

This thesis first proposes a fast online Support Vector Machine algorithm (LASVM) that has an outstanding speed improvement over the classical (batch)

SVM and other online SVM algorithms, while preserving the classification accuracy rates of the state-of-the-art SVM solvers. The ability to handle streaming data, speed improvement in both training and recognition and the demand for less memory with the online learning setting enable SVM to be applicable to very large datasets. The effectiveness of LASVM and active learning in real world problems is assessed by targeting the name disambiguation problem in CiteSeer's repository of scientific publications. The thesis then presents an efficient active learning framework to target the expensive labeling problem for producing training data. The proposed method yields an efficient querying system and removes the barriers of applying active learning to very large scale datasets due to the high computational costs. We then point out that even when the labels are readily available, active learning can still be used to reach out to the most informative instances in the training data. By applying active sample selection and early stopping in the online SVM, we show that the algorithm can reach and even exceed the prediction accuracies of baseline setting of LASVM with random sampling. Our experimental results also reveal that active learning can be a highly effective method for dealing with the class imbalance problem. We further propose a hybrid method of oversampling and active learning to form an adaptive technique (named VIRTUAL) to efficiently resample the minority class instances in imbalanced data classification. Finally, we propose a non-convex online SVM algorithm (LASVM-NC) based on the *Ramp loss*, which has strong ability of suppressing the influences of outliers in noisy datasets. Then, again in the online learning setting, we propose an outlier filtering mechanism that approximates non-convex behavior in convex optimization (LASVM-I). These two algorithms are built on an online SVM solver (LASVM-G)

which leverages the duality gap to obtain more trustworthy intermediate models. Our experimental results show that the proposed approaches yield a more scalable online SVM algorithm with sparser models and less computational running time both in the training and recognition phases without sacrificing generalization performance. In the end, we also point out the relation between the non-convex behavior in SVMs and active learning.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I feel privileged for having a chance to pursue my Ph.D studies at Penn State, home to one of the best and largest engineering colleges in the nation. I cannot imagine having experienced a more challenging, fulfilling, or inspiring Ph.D experience anywhere else. During my graduate studies, I have received inspiration, encouragement and support from a number of individuals and institutions. This thesis would not have taken its current form without their contributions.

I had the distinct opportunity to be a member of the world-renowned CiteSeer lab at Penn State. First and foremost, I would like to extend my deepest thanks and appreciation to my research advisor, Prof. C. Lee Giles, for providing such a stimulating research environment, offering opportunities for collaborations and helping me to become forward-thinking and self sufficient. I am especially grateful for his excellent advice, direction and suggestions, the enthusiasm he expressed for my work and the flexibility and freedom he provided me to explore new avenues of research. I am also very grateful to my supervisor Dr. Léon Bottou at NEC Research Labs for having provided me with his expertise, for introducing me to the magical world of machine learning and being very instrumental in developing my knowledge of and appreciation for the study of machine learning. Discussions with him have been a source of tremendous insights, and observing him in action a source of inspiration. His enthusiasm for fundamental research is admirable. His guidance, suggestions and comments made me a better researcher. I am truly indebted to both of my advisors and grateful for the unique and rewarding opportunity of working with them for many years.

My research and my thesis greatly benefited from the useful suggestions, comments, guidance and feedback from the members of my exceptional doctoral committee, and I wish to thank Professors Raj Acharja, Jia Li and Tracy Mullen for all their contributions.

Special thanks go to Prof. Brian Cameron for providing me with the opportunity to work on interesting projects related to my research and advancing my understanding to solve real world problems through consultancy positions with his

industry affiliates.

I am also indebted to Eric Glover for helping me establish my research topics in the early stages of my Ph.D studies and initiating my affiliation with the NEC Research Labs. I would like to thank NEC Research Labs for partially sponsoring my Ph.D research and providing me with the computing resources to run my large scale experiments. At NEC, I also had the opportunity to collaborate with many distinguished scientists, including Vladimir Vapnik, Hans Peter Graf, Eric Cosatto, Jason Weston and Ronan Collobert, whose combination of intellectual rigor and scientific curiosity has shaped my attitude towards research and will continue to be a source of inspiration in my future endeavours.

My time in graduate school would not have been the same without the collaborations, stimulating exchanges and laughs provided by other members of the Intelligent Information Systems Lab. Thank you all for the fun memories. I especially would like to thank Jian Huang for fruitful discussions and collaborations, and Isaac Councill for sharing his technical expertise and for making the late night studies more productive.

I would like to thank the administrative staff of the computer science department, especially Vicki Keller, for totally easing the burden of bureaucracy and helping us in the fastest way in all kinds of administrative tasks.

My research has greatly benefited from countless discussions at seminars, workshops and conferences. I am grateful to NSF, ACM, Google, Grace Hopper Consortium and Penn State's Office of Graduate Studies, Research & Outreach (OGSRO) and Women in Engineering Program (WEP) for granting me conference scholarships and travel awards during my graduate studies. I have significantly benefited from attending various conferences to present my work and to build connections with other researchers.

My deepest gratitude goes to my parents, Ayşe Ertekin and Sadullah Ertekin, who not only gave me a wonderful life but also empowered me with the skills, mindset and drive for the things that I want to accomplish in life. Thank you Mom and Dad for teaching me to be brave in the face of adversity, strong during times of challenge and determined in identifying my goals in life. Thank you for being the wind under my wings in all my scholarly and personal endeavors. Mom, I certainly feel one of the luckiest people on earth for not only having a wonderful mother like you but also having you as my school teacher for five years. You extended your own creativity and love of learning to me when I was young. Thank you for giving me a strong foundation to build my life on, teaching the value of hard work and instilling in me the dedication necessary for completing intensive projects like this. Dad, you always were, and always will remain my hero for being a man of integrity, a smart and talented engineer and most importantly a great father. Your exquisite talent in analytical thinking and technical creativity

has always been my inspiration. It was one of my most joyful and memorable moments to play in your chemistry lab with my brother when we were very little. Thank you for exposing me to the exciting world of engineering and always being there for me in realizing my dreams. Oğuz, dear brother of mine, you will always remain my little brother and will always be special to me. Mom and Dad, once again, thank you for your constant support and endless love and teaching me the importance and distinction of making a difference in life.

Last and most of all, I want to express my sincere gratitude to my beloved husband Dr. Levent Bolelli, an accomplished computer scientist, an extremely talented engineer and most importantly a loving husband and a father. Thank you for not only providing me with motivation, understanding, support and encouragement at every step of this pursuit but also for being a great collaborator at work. You are the most important factor for my well being and achievements in life. Words alone could never convey the depth of my sincere love and appreciation for you. Thank you for being, together with our baby, the greatest happiness in my life.

Many thanks go to Alya as well, the best baby a mother could have, for becoming the meaning of my life. My sweet baby, thank you for giving me the opportunity to get some writing done during your afternoon naps, for only tearing up my papers and throwing them on the floor a few times, for being there to play with when I did not feel like working and for finding new ways every day to bring joy and happiness to my life. I am extremely grateful for having you as my child and I am already proud of you.

The completion of this thesis represents the culmination of a tremendous amount of work, effort and knowledge. Levent, we have completed one more milestone in our lives. As it echoed in space in one of the moon landings, *it has been a long way but we are here*. And as you know baby, this is not an end; this is only the beginning!

Dedication

To the best parents ever

Ayşe Ertekin and ***Sadullah Ertekin***,

to my husband, my love

Levent,

and to my shining star, *bebeğim*, my everything

Alya Su.

Chapter 1

Introduction

"Başlamak, bitirmenin yarısıdır."

Türk Atasözü

"To begin is half the work."

A Turkish Proverb

In the last decades, humankind has been witnessing an information revolution that can be traced back to the invention of integrated circuits and computer chips. Today, we are surrounded by ubiquitous electronic devices and computing resources that have penetrated every aspect of our lives. These advances in computer and information technology have created a profound effect on our lives that can only be paralleled by the scientific revolution that happened centuries ago and the subsequent industrial revolution. The clearly stated fundamental theories, augmented with the systematic collection and scientific analysis of data, enabled the scientific revolution to establish a base for many modern sciences and to increase our understanding of the universe, life, matter and society. At a later period, the industrial revolution marked fundamental changes in many areas of human life,

including manufacturing, production, communication and transportation. These periods were major turning points in human society and almost every aspect of daily life was eventually influenced in some way. Today, we are witnessing changes of even larger magnitude. We are surrounded by vast amount of data produced by various sources, such as sensors, cameras, microphones, computer systems, industrial machines and humans. The size of the discoverable Web has grown to tens of billions of web pages and has been one of the main sources of our information and knowledge. Tremendous amount of data is being gathered in sensor networks for monitoring, tracking, and controlling purposes in various applications, such as traffic monitoring, habitat monitoring, nuclear reactor control and object tracking. Advances in electronics and computing technologies altered the manner in which we work, entertain, trade, communicate and collaborate with each other. These advances also allowed us to collect, analyze and visualize vast amount of data in numerous fields, including astronomy, oceanography, business, economics, finance, medicine and biology, that was beyond our processing capabilities before. Therefore, science in the 21^{st} century requires a different set of computational techniques and algorithms which will allow us to tackle the problems and challenges in large scale data analysis.

We are observing a paradigm shift in the fundamental principles of scientific modeling in many fields of modern science and engineering. Historically, our focus has been centered around *first-principle models* to describe real world phenomena. That is, we start with a basic scientific model (e.g, Newton's laws of mechanics or Maxwell's theory of electromagnetism), fit experimental data (measurements) to the defined model, and then estimate the model parameters that can't be measured directly. While this approach has been suitable for a wide range of applications, the characteristics and scale of numerous scientific problems of today prevents us

from using first-principle models [1]. For instance, the first principles may not be applied in situations when human expertise is absent (i.e. navigating on Mars) or can't be explained (i.e. speech recognition, vision), the solution changes over time (i.e. portfolio management, routing in a computer network) or needs to be adaptive (i.e. user biometrics, personalized search), or the size of the problem is too vast for our limited reasoning capabilities (i.e. calculating the ranks of webpages in search engines, analyzing data collected by space telescopes, mining historical medical records, understanding customer behavior from transactions). Such constraints has driven the paradigm shift in computational modeling, and consequently, our focus has changed to a data-centered view. In the absence of first-principle models, the vast amount of data becomes the source of the models by estimating useful relationships between a system's variables. In short, we are shifting our focus from fitting data into predefined models to finding models that can best describe the data.

Machine learning takes center stage in our quest for making sense of the vast amount of data by turning it into information and turning information into knowledge. Machine learning involves techniques to allow computers to "learn". Specifically, machine learning involves *training* a computer system to perform some task, rather than directly *programming* the system to perform the task. Machine learning systems observe some data (i.e. training data) and automatically determine some characteristics of the data to use at a later time when processing unknown data (i.e. test data). The training data consists of pairs of input objects, and desired outputs. The output of the function can be a continuous value (i.e. regression) or can predict a class label of the input object (i.e. classification). The task of the learning machine is to predict the value of the function for any valid input object after having seen only a small number of training examples.

1.1 Problem Statement

Applying machine learning algorithms to very large scale problems still poses challenges. Three factors limit machine learning algorithms when both the sample size n and the input dimension d grows very large. First, labeling of the training examples becomes very expensive. Second, training time becomes unreasonably long. Third, the size of the model grows, which affects the memory requirements during both training and recognition phases. Moreover, regardless of the sample size and input dimension, training data can be imbalanced and noisy, which degrades the generalization performance of learning algorithms. These problems collectively are related to the two main components of any learning problem: the scalability of learning algorithms and the characteristics of the datasets. The rest of this section further elaborates on each component, and the next section highlights our contributions.

Historically, the main goal of machine learning research has been to "get it right". That is, most of the focus has been on developing algorithms that have better generalization properties and yielding higher accuracy on unseen data. While we believe that we should never lose sight of this goal, we note that the computational requirements and scalability of learning algorithms often become an afterthought. With the ever increasing rate at which we generate, gather and store data, it has become evident that computing cost of algorithms is a major concern in dealing with large scale datasets. For instance, we can not benefit much from a classification algorithm that (*i*) requires more computational resources than we can provide, (*ii*) is unreasonably slow to train a model, or (*iii*) takes very long to make decisions on new observations. Hence, we need efficient algorithms that can respond to the requirements of learning from large scale datasets. These require-

ments include obtaining labels of training examples and reaching out to the most informative instances in the training data in a cost efficient way, training models in reasonable time, and building sparser and simpler models that use less memory in training and recognition phases.

Aside from the quantity of the data, the characteristics of datasets also pose challenges to learning algorithms, and their effects are exacerbated in large scale datasets. These problems mostly present themselves as noise in the input data or uneven distributions of classes. The existence of noise in datasets, resulting from unreliable labeling sources and/or the characteristics of the dataset or domain, may cause overfitting and reduce the classification performance of learning algorithms. Furthermore, they become a drag on the efficiency of learners by slowing down the training and prediction processes due to increased model sizes. In addition to the problems caused by noise in datasets, class imbalance problem also has been known to hinder the learning performance of classification algorithms. The problem occurs when there are significantly less number of examples of the target class. Real world applications often face this problem because naturally, normal examples which constitute the majority class are generally abundant but the examples of interest that form the minority class are generally rare, costly or difficult to collect. Due to their design principles, most machine learning algorithms optimize the overall classification accuracy, therefore the learned class boundary can be severely skewed toward the minority class. Consequently, the failure rate in detecting the instances of the minority class can be excessively high.

In this thesis, we present algorithms and methodologies to overcome these aforementioned issues that stem from the *quantity*, *quality* and the *distribution* of data in machine learning problems. The main focus of this thesis is novel machine learning algorithms and approaches that are faster, data efficient and less demanding in

computational resources to achieve scalable algorithms for large scale data mining problems. In the next section, we outline our contributions.

1.2 Contributions of This Thesis

The exponential growth of the amount of digital information has significantly increased the need to access right and relevant information and the desire to organize and categorize data. Therefore, automatic classification, – the assignment of instances (i.e., pictures, text documents, DNA sequences, Web sites) to one or more predefined categories based on their content has become a very important field of machine learning research. One popular machine learning algorithm for automatic data classification is Support Vector Machines (SVM), due to its strong theoretical foundation and good generalization performance. However, SVM did not see widespread adoption in communities that work with very large datasets because of the high computational cost involved in solving the quadratic programming problem in the training phase of the learner. This thesis addresses the scalability problem and the issues that stem from class imbalance and noisy data in SVM. Specifically, we propose algorithms and approaches that enable SVM to (*i*) scale to very large datasets with online and active learning, (*ii*) yield improved computational efficiency and prediction performance in classification of imbalanced datasets, and (*iii*) achieve faster learning and sparser solutions without sacrificing classification accuracy in noisy datasets by showing the benefits of using non-convex optimization for online SVM.

This thesis does not attempt to solve a specific problem in a particular domain (i.e. text, image, speech), but addresses the problem of *computational learning* in a general way with the ultimate goal of *generalization* with sparse models and

scalable solutions. The particular focus is on Support Vector Machine algorithm with classification problems, but the proposed approaches on active and online learning are also well extensible to other widely used machine learning algorithms for various tasks. Several application areas may benefit from the methods and approaches proposed in this thesis: Medical and health care informatics, computational medicine, web page categorization, decision automation systems in the business and engineering world, machine fault detection in industries, analyzing space telescope data in astronomy are only some of the examples.

LASVM – An Online SVM Algorithm for Massive and Streaming Data: The sizes of the datasets are quickly outgrowing the computing power of our computers. If we look at the advances in computer hardware technology in the last decade, hard disk capacities became thousand times larger but processors became only hundred times faster. Therefore, we need faster machine learning algorithms in order to make computers learn more efficiently from experimental and historical data. Moreover, in many domains, data now arrives faster than we are able to learn from it. To avoid wasting this data, we must switch from the traditional *batch* machine learning algorithms to *online* systems that are able to mine streaming, high-volume, open-ended data as they arrive. We present a fast online Support Vector Machine classifier algorithm, namely LASVM [2] that can handle continuous stream of new data and has an outstanding speed improvement over the classical (batch) SVM and other online SVM algorithms, while preserving the classification accuracy rates of the state-of-the-art SVM solvers. The speed improvement and the demand for less memory with the online learning setting enable SVM to be applicable to very large datasets. As an application to a real world system, we developed a name disambiguation framework for CiteSeer that

utilizes LASVM as a distance function to determine the similarity of different author metadata records and a clustering step that determines unique authors based on LASVM-based similarity metric. Applied to the entire CiteSeer repository with more than 700,000 papers, the algorithm efficiently identified and disambiguated close to 420,000 unique authors in CiteSeer.

Active Learning and VIRTUAL for Class Imbalance Problem: The class imbalance problem occurs when there are significantly less number of observations of the target concept. However, standard machine learning algorithms yield better prediction performance with balanced datasets. We demonstrate that active learning is capable of solving the class imbalance problem by providing the learner more balanced classes [3]. The proposed method also yields an efficient querying system and removes the barriers for applying active learning to very large scale datasets. With *small pool active learning* [3], we show that we do not need to search the entire dataset to find the informative instances. We also propose a hybrid algorithm, called VIRTUAL (**V**irtual **I**nstances **R**esampling **T**echnique **U**sing **A**ctive **L**earning) [4], that integrates oversampling and active learning methods to form an adaptive technique for resampling of minority class instances. VIRTUAL is more efficient in generating new synthetic instances and has a shorter training time than other oversampling techniques due to its adaptive nature and its efficient decision capability in creating virtual instances.

Online Non-Convex Support Vector Machines: Databases often contain noise in the form of inaccuracies, inconsistencies and false labeling. Noise in the data is notorious for degrading the prediction performance and computational efficiency of machine learning algorithms. We design an online non-convex

Support Vector Machine algorithm (LASVM-NC) [5], which has strong ability of suppressing the influences of outliers (mislabeled examples). Then, again in the online learning setting, we propose an outlier filtering mechanism based on approximating non-convex behavior in convex optimization (LASVM-I) [5]. These two algorithms are built upon another novel SVM algorithm (LASVM-G) [5] that is capable of generating accurate immediate models in its iterative steps by leveraging the duality gap. We argue that despite many advantages of convex modeling, the price we pay for insisting on convexity is an increase in the size of the model and the scaling properties of the algorithm. We show that shifting gears from convexity to non-convexity can be very effective for achieving sparse and scalable solutions, particularly when the data consists of abundant label noise.

1.3 Organization of the Thesis

The rest of this thesis is organized as follows: In Chapter 2, we present an overview of Support Vector Machines. Chapter 3 presents LASVM, an efficient online SVM solver that can handle continuous stream of new data and is highly scalable to large scale datasets. Chapter 4 presents a small pool active learning methodology, which makes classical active learning computationally efficient and applicable to large scale datasets without sacrificing prediction accuracy. This chapter also presents that active learning is highly effective to address the class imbalance problem. In Chapter 5, we propose a computationally efficient, adaptive oversampling method VIRTUAL that creates virtual instances for the minority class support vectors which yields competitive and even higher prediction accuracies in imbalanced data classification. Chapter 6 first presents LASVM-G – an efficient online SVM that bring performance enhancements to LASVM by leveraging the duality gap. The

chapter then introduces LASVM-NC – a non-convex online SVM solver which yields significant speed-up in learning from noisy datasets by building sparser SVM models. The chapter continues with introducing LASVM-I – A convex SVM solver that approximates LASVM-NC by an outlier suppression mechanism. Concluding remarks and directions for future research are outlined in Chapter 7.

Chapter 2

Support Vector Machines

"There is nothing more practical than a good theory."

Kurt Lewin

The main objective of statistical learning is to find a description of an unknown dependency between measurements of objects and certain properties of these objects. The measurements, also known as "input variables", are assumed to be observable in all objects of interest. On the contrary, the properties of the objects, or "output variables", are in general available only for a small subset of objects. The purpose of estimating the dependency between the input and output variables is to be able to determine the values of output variables for any object of interest. In pattern recognition, this relates to trying to estimate a function $f : \mathbb{R}^N \mapsto \{\pm 1\}$ that can correctly classify new examples based on past observations.

Support Vector Machines [6] are one of the most popular classification algorithms to perform such tasks and are well known for their strong theoretical foundations, generalization performance and ability to handle high dimensional data. This section presents an overview of support vector learning, starting with linear

SVMs, followed by their extension to the nonlinear case.

2.1 Linear SVMs

2.1.1 Separable Case

In the binary classification setting, let $((x_1, y_1) \cdots (x_n, y_n))$ be the training dataset where x_i are the feature vectors representing the instances (i.e. observations) and $y_i \in (-1, +1)$ be the labels of the instances. Support vector learning is the problem of finding a *separating hyperplane* that separates the positive examples (labeled +1) from the negative examples (labeled -1) with the largest margin. The margin of the hyperplane is defined as the shortest distance between the positive and negative instances that are closest to the hyperplane. The intuition behind searching for the hyperplane with a large margin is that a hyperplane with the largest margin should be more resistant to noise than a hyperplane with a smaller margin.

Formally, suppose that all the data satisfy the constraints

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq +1 \quad y_i = +1 \tag{2.1}$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1 \quad y_i = -1 \tag{2.2}$$

where $\boldsymbol{w}$ is the normal to the hyperplane, $|b|/\|\boldsymbol{w}\|$ is the perpendicular distance from the hyperplane to the origin, and $\|\boldsymbol{w}\|$ is the Euclidean norm of $\boldsymbol{w}$. These two constraints can be conveniently combined into the following

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \quad \forall i \tag{2.3}$$

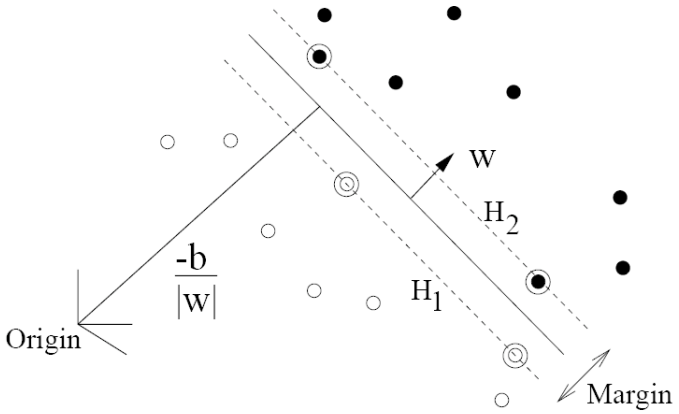


Figure 2.1. A hyperplane separating two classes with the maximum margin. The circled examples that lie on the *canonical hyperplanes* are called support vectors.

The training examples for which (2.3) holds lie on the canonical hyperplanes (H_1 and H_2 in Figure 2.1). The margin ρ can then be easily computed as the distance between H_1 and H_2.

$$\rho = \frac{|1-b|}{\|\boldsymbol{w}\|} - \frac{|-1-b|}{\|\boldsymbol{w}\|} = \frac{2}{\|\boldsymbol{w}\|} \tag{2.4}$$

Hence, the maximum margin separating hyperplane can be constructed by solving the following *Primal* optimization problem

$$\min_{\boldsymbol{w}\in\mathcal{H}} \tau(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{subject to} \quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 \quad \forall i \tag{2.5}$$

We switch to a Lagrangian formulation of this problem for two main reasons: i) the constraints are easier to handle, and ii) training data only appears as a dot product between vectors. This formulation introduces a new Lagrange multiplier α_i for each constraint and the formulation of the minimization problem then becomes,

$$\min_{\boldsymbol{w},b} \; L(\boldsymbol{w}, b, \alpha) \equiv \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\boldsymbol{x}_i \boldsymbol{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{2.6}$$

with Lagrange multipliers $\alpha_i \geq 0$ for each constraint in (2.5). The objective is then to minimize (2.6) with respect to $\boldsymbol{w}$ and b and simultaneously require that the derivatives of $L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ with respect to all the $\boldsymbol{\alpha}$ vanish.

2.1.1.1 The Karush-Kuhn-Tucker Conditions

The Karush-Kuhn-Tucker (KKT) [7, 8] conditions establish the requirements that need to be satisfied by an optimum solution to a general optimization problem. Given the primal problem in (2.6), KKT conditions state that the solutions $\boldsymbol{w}^*, b^*$ and α^* should satisfy the following conditions

$$\frac{\partial L(\boldsymbol{w}^*, b^*, \alpha^*)}{\partial \boldsymbol{w}} = w_v - \sum_i \alpha_i y_i x_{iv} = 0 \quad v = 1, \cdots, d \tag{2.7}$$

$$\frac{\partial L(\boldsymbol{w}^*, b^*, \alpha^*)}{\partial b} = -\sum_i \alpha_i y_i = 0 \tag{2.8}$$

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0, \quad \forall i \tag{2.9}$$

$$\alpha_i \geq 0 \quad \forall i \tag{2.10}$$

$$\alpha_i(y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - 1) = 0 \quad \forall i \tag{2.11}$$

The problem for SVM is convex and thus, these KKT conditions are *necessary* and *sufficient* for $\boldsymbol{w}^*, b^*, \alpha^*$ to be a solution [9]. Therefore, solving the SVM problem is equivalent to finding a solution to the KKT conditions. The first KKT condition defines the optimal hyperplane as a linear combination of the vectors in the training set

$$\boldsymbol{w}^* = \sum_i \alpha_i^* y_i \boldsymbol{x}_i \tag{2.12}$$

whereas the second KKT condition requires that the α_i coefficients of the training instances should satisfy

$$\sum_{i=1}^{n} \alpha_i^* y_i = 0 \ , \quad \alpha_i^* \geq 0 \tag{2.13}$$

2.1.1.2 The Dual Problem

In practice, the SVM problem is solved through its "dual" definition: *Maximize* (2.6) with respect to $\boldsymbol{\alpha}$, and minimize with respect to $\boldsymbol{w}$ and b. At this saddle point, the derivatives of L with respect to $\boldsymbol{w}$ and b must vanish following the KKT conditions. Substituting (2.7) and (2.8) into (2.6) gives us the formulation of the dual

$$\max_{\alpha} L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \quad \text{subject to} \quad \forall i \begin{cases} \sum_i \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{cases} \tag{2.14}$$

Thus, by solving the dual optimization problem, one obtains the coefficients α_i. The instances with $\alpha_i > 0$ are called "support vectors" and they lie on either of the canonical hyperplanes H_1 or H_2. Note that only the instances with $\alpha_i > 0$ effect the SVM solution and thus, only the support vectors are needed to express the solution $\boldsymbol{w}$. This leads to the decision function

$$\begin{aligned} f(\boldsymbol{x}) &= \boldsymbol{w}^T \boldsymbol{x}_i + b \\ &= \sum_{i=1}^{M} y_i \alpha_i (\boldsymbol{x}_i^T \boldsymbol{x}) + b \end{aligned} \tag{2.15}$$

The sign of the decision function determines the predicted classification of $\boldsymbol{x}$.

2.1.2 Non-Separable Case

The previous section discussed the case where it is possible to linearly separate the training instances that belong to different classes. Obviously this SVM formulation will not find a solution if the data can not be separated by a hyperplane. Even in the cases where the data *is* linearly separable, SVM may overfit to the training data in its search for the hyperplane that completely separates all of the instances of both classes. For instance, an individual outlier in a dataset, such as a pattern which is mislabeled, can crucially affect the hyperplane. These concerns prompted the development of *soft margin* SVMs [10], which can handle linearly non-separable data by introducing positive *slack variables* ξ_i that relax the constraints in (2.1) and (2.2) at a cost proportional to the value of ξ_i. Based on this new criteria, the relaxed constraints with the slack variables then become

$$\forall i \quad \begin{cases} \boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq +1 - \xi_i & y_i = +1 \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1 - \xi_i & y_i = -1 \\ \xi_i \geq 0. \end{cases} \tag{2.16}$$

which permits some instances to lie inside the margin or even cross further among the instances of the opposite class (see Figure 2.2). While this relaxation gives SVMs flexibility to decrease the influence of outliers, from an optimization perspective, it is not desirable to have arbitrarily large values for ξ_i as that would cause the SVM to obtain trivial and sub-optimal solutions. Thus, we "constrain the relaxation" by making the slack variables part of the objective function (2.5), yielding

$$\min_{\boldsymbol{w} \in \mathcal{H}, \boldsymbol{\xi} \in \mathbb{R}^m} \tau(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^{m} \xi_i \tag{2.17}$$

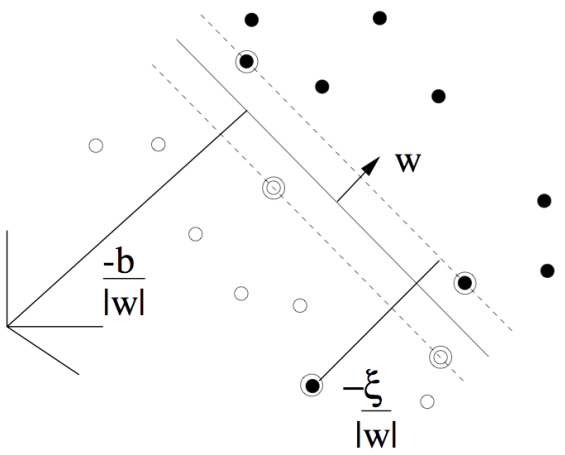


Figure 2.2. Soft margin SVM.

subject to the constraints in (2.16). The cost coefficient $C > 0$ is a hyperparameter that specifies the misclassification penalty and is tuned by the user based on the classification task and dataset characteristics.

As in the separable case, the solution to (2.17) can be shown to have an expansion

$$\frac{\partial L_P}{\partial \boldsymbol{w}} = 0 \quad \Leftrightarrow \quad \boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i \tag{2.18}$$

where the training instances with $\alpha_i > 0$ are the support vectors of the SVM solution. Note that the penalty function related to the slack variables is linear, which disappears when we transforming (2.17) into the dual formulation

$$\max_{\alpha} L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \quad \text{subject to} \quad \forall i \begin{cases} \sum_i \alpha_i y_i = 0 \\ C \leq \alpha_i \leq 0 \end{cases} \tag{2.19}$$

The dual formulation is conveniently very similar to the linearly separable case, with the only difference being the extra upper bound of C for the coefficients α_i. Obviously as the misclassification penalty $C \to \infty$, (2.19) converges to the linearly separable case.

2.2 Nonlinear Support Vector Machines

Until now, we discussed SVMs in the specific context of learning linear decision boundaries. However, the power of SVMs can be fully realized when we extend the linear SVM to allow more general decision surfaces. Remember that one of the two benefits of Lagrangian transformation of the SVM problem was that the training data appears as a dot product between vectors (see Eq. 2.19). We can exploit this fact to use "kernel trick" to allow SVM to form nonlinear boundaries.

Suppose we map the training data to some other Euclidean space $\mathcal{H}$ using a mapping $\Phi : \mathbb{R}^d \mapsto \mathcal{H}$. From the training algorithm's point of view, the only affect of this transformation is that, instead of computing the dot products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ in $\mathbb{R}^d$, the algorithm would now depend on the data through the dot products in $\mathcal{H}$, i.e. on functions of the form $\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$. Note that we don't necessarily need to perform the computationally expensive Φ transformation for each example. If there were a kernel function K that corresponds to a dot product in some expanded feature space (i.e. $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)$), we would only need to compute the value of K to find the dot products in Φ, without even mapping $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ from $\mathbb{R}^d$ to $\mathcal{H}$. In the dual problem in (2.19), we then replace the dot product with the new kernel function K that corresponds to a dot product for the transformation Φ.

$$\max_{\alpha} L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \quad \text{subject to} \quad \forall i \begin{cases} \sum_i \alpha_i y_i = 0 \\ C \leq \alpha_i \leq 0 \end{cases} \tag{2.20}$$

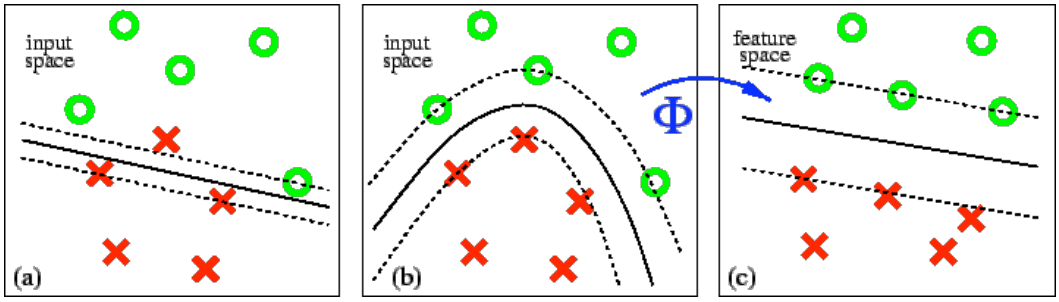


Figure 2.3. Effect of kernel transformation. The data is not linearly separable in the input space (a), and the data can only be separated by a non-linear surface (b). This non-linear surface in the input space corresponds to a linear surface in a kernel-mapped feature space.

2.2.1 Kernel Functions

The basic idea with nonlinear SVMs is to map training data into a higher dimensional *feature* space via some mapping $\Phi(x)$ and construct a separating hyperplane with maximum margin in the input space. As shown in Figure 2.3, the linear decision function in the feature space corresponds to a non-linear decision boundary in the original input space. By use of a kernel function, $K(x, y) = \langle \Phi(x), \Phi(y) \rangle$, it is possible to compute the separating hyperplane without explicitly carrying out the mapping into feature space. To ensure that a kernel function actually corresponds to some feature space, it must be symmetric, i.e. $K(x, y) = \langle \Phi(x), \Phi(y) \rangle = \langle \Phi(y), \Phi(x) \rangle = K(y, x)$. Typically, kernels need to satisfy Mercer's Theorem, which states that the kernel matrix $\mathbf{K} = (K(x_i, x_j))_{i,j=1}^{n}$ must be positive semi-definite, i.e. it has no non-negative eigenvalues. Typical choice for kernels are

- Linear Kernel: $K(x, y) = \langle x, y \rangle$
- Polynomial Kernel: $K(x, y) = (\langle x, y \rangle)^d$
- RBF Kernel: $K(x, y) = \exp(\frac{-\|x-y\|^2}{2\sigma^2})$
- Sigmoid Kernel: $K(x, y) = \tanh(\gamma \langle x, y \rangle - \theta)$

Each kernel corresponds to some feature space and because no explicit mapping to this feature space occurs, optimal linear separators can be found efficiently in feature spaces with millions of dimensions.

2.3 Quadratic Programming Solvers for SVMs

In this section, we present algorithms for solving the SVM QP problem in (2.20). In our discussions, we use a slight variation of the SVM dual problem and adopt the following representation for the constraints

$$\max_{\boldsymbol{\alpha}} \; L_D \quad \text{with} \quad \begin{cases} \sum_i \alpha_i = 0 \\ A_i \le \alpha_i \le B_i \\ \quad A_i = \min(0, Cy_i) \\ \quad B_i = \max(0, Cy_i) \end{cases} \tag{2.21}$$

The above formulation slightly deviates from the standard formulation [10] because it makes the α_i coefficients positive when $y_i = +1$ and negative when $y_i = -1$.

2.3.1 Sequential Direction Search

Efficient numerical algorithms have been developed to solve the SVM QP problem (2.21). The best known methods are the Conjugate Gradient method [11, pages 359–362] and the Sequential Minimal Optimization [12]. Both methods work by making successive searches along well chosen directions.

Each direction search solves the restriction of the SVM problem to the half-line starting from the current vector $\boldsymbol{\alpha}$ and extending along the specified direction $\boldsymbol{u}$.

Such a search yields a new feasible vector $\boldsymbol{\alpha} + \lambda^* \boldsymbol{u}$.

$$\lambda^* = \arg\max W(\boldsymbol{\alpha} + \lambda \boldsymbol{u}) \quad \text{with} \quad 0 \leq \lambda \leq \phi(\boldsymbol{\alpha}, \boldsymbol{u}) \tag{2.22}$$

The upper bound $\phi(\boldsymbol{\alpha}, \boldsymbol{u})$ ensures that $\boldsymbol{\alpha} + \lambda \boldsymbol{u}$ is feasible as well.

$$\phi(\boldsymbol{\alpha}, \boldsymbol{u}) \;=\; \min \left\{ \begin{array}{ll} 0 & \text{if} \quad \sum_k u_k \neq 0 \\ (B_i - \alpha_i)/u_i & \text{for all } i \text{ such that } u_i > 0 \\ (A_j - \alpha_j)/u_j & \text{for all } j \text{ such that } u_j < 0 \end{array} \right\} \tag{2.23}$$

Calculus shows that the optimal value is achieved for

$$\lambda^* = \min \left\{ \; \phi(\boldsymbol{\alpha}, \boldsymbol{u}) \; , \; \frac{\sum_i g_i \, u_i}{\sum_{i,j} u_i u_j \, K_{ij}} \; \right\} \tag{2.24}$$

where $K_{ij} = K(x_i, x_j)$ and $\boldsymbol{g} = (g_1 \ldots g_n)$ is the gradient of $W(\boldsymbol{\alpha})$.

$$g_k \;=\; \frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_k} \;=\; y_k - \sum_i \alpha_i K(x_i, x_k) \;=\; y_k - \hat{y}(x_k) + b \tag{2.25}$$

2.3.2 Sequential Minimal Optimization

Direction search computations are much faster when the search direction $\boldsymbol{u}$ mostly contains zero coefficients [12]. At least two coefficients are needed to ensure that $\sum_k u_k = 0$. The *Sequential Minimal Optimization* (SMO) algorithm uses search directions whose coefficients are all zero except for a single $+1$ and a single -1.

Practical implementations of the SMO algorithm [13, 14] usually rely on a small positive tolerance $\tau > 0$. They only select directions $\boldsymbol{u}$ such that $\phi(\boldsymbol{\alpha}, \boldsymbol{u}) > 0$ and $\boldsymbol{u}'\boldsymbol{g} > \tau$. This means that we can move along direction $\boldsymbol{u}$ without immediately reaching a constraint and increase the value of $W(\boldsymbol{\alpha})$. Such directions are defined

by the so-called τ-*violating pair* (i, j):

$$(i,j) \text{ is a } \tau\text{-violating pair} \iff \begin{cases} \alpha_i < B_i \\ \alpha_j > A_j \\ g_i - g_j > \tau \end{cases}$$

SMO Algorithm

1) Set $\boldsymbol{\alpha} \leftarrow \mathbf{0}$ and compute the initial gradient $\boldsymbol{g}$ (equation 2.25)

2) Choose a τ-violating pair (i, j). Stop if no such pair exists.

3) $\lambda \leftarrow \min \left\{ \dfrac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}},\ B_i - \alpha_i,\ \alpha_j - A_j \right\}$

   $\alpha_i \leftarrow \alpha_i + \lambda\ , \quad \alpha_j \leftarrow \alpha_j - \lambda$

   $g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall\, s \in \{1 \ldots n\}$

4) Return to step (2)

The above algorithm does not specify how exactly the τ-violating pairs are chosen. Modern implementations of SMO select the τ-violating pair (i, j) that maximizes the directional gradient $\boldsymbol{u}'\boldsymbol{g}$. This choice was described in the context of Optimal Hyperplanes in both [11, pages 362–364] and [15].

Regardless of how exactly the τ-violating pairs are chosen, the SMO algorithm stops after a finite number of steps [16]. When SMO stops, no τ-violating pair remain. The corresponding $\boldsymbol{\alpha}$ is called a τ-*approximate solution*.

Note from (2.8) that the constraint on the sum of α_i in (2.21) appears in the SVM problem only when we allow the offset (bias) term to exist (i.e. $b \neq 0$). While there is a single "optimal" b, different SVM implementations may choose a separate way of adjusting the offset. For instance, it is sometimes useful change b in order to adjust the number of false positives and false negatives [17, page 203], or even disallow the bias term completely (i.e. set $b = 0$) [18] for compu-

tational simplicity, which removes the constraint (2.8). SMO works on SVM QP problems *with* the constraint (2.8). In SVM implementations which disallow the offset term, we don't need to use SMO since the constraint on the sum of α's is removed from the SVM problem. Therefore, the algorithm achieves flexibility to update a single α_i at a time at each optimization step, bringing further computational simplicity and efficiency to the solution of the SVM problem. In Chapter 2, we present an efficient online SVM solver, LASVM, that formulates the SVM problem with the offset term b. Chapter 6 introduces a non-convex online SVM algorithm that formulates the SVM optimization problem without the offset b to achieve computational advantages without sacrificing competitive generalization performance.

2.4 Computational cost of SVMs

There are two intuitive lower bounds on the computational cost of any algorithm able to solve the SVM QP problem for arbitrary matrices $K_{ij} = K(x_i, x_j)$.

1. Suppose that an oracle reveals whether $\alpha_i = 0$ or $\alpha_i = \pm C$ for all $i = 1 \dots n$. Computing the remaining $0 < |\alpha_i| < C$ amounts to inverting a matrix of size $R \times R$ where R is the number of support vectors such that $0 < |\alpha_i| < C$. This typically requires a number of operations proportional to R^3.

2. Simply verifying that a vector $\boldsymbol{\alpha}$ is a solution of the SVM QP problem involves computing the gradients of $W(\boldsymbol{\alpha})$ and checking the KKT optimality conditions. With n examples and S support vectors, this requires a number of operations proportional to $n\,S$.

Few support vectors reach the upper bound C when it gets large. The cost is then dominated by the $R^3 \approx S^3$. Otherwise the term nS is usually larger. The final number of support vectors therefore is the critical component of the computational cost of the SVM QP problem.

Assume that increasingly large sets of training examples are drawn from an unknown distribution $P(x, y)$. Let $\mathcal{B}$ be the error rate achieved by the best decision function (2.15) for that distribution. When $\mathcal{B} > 0$, Steinwart [19] shows that the number of support vectors is asymptotically equivalent to $2n\mathcal{B}$. Therefore, regardless of the exact algorithm used, the asymptotical computational cost of solving the SVM QP problem grows at least like n^2 when C is small and n^3 when C gets large. Empirical evidence shows that modern SVM solvers [13, 14] come close to these scaling laws.

Practice however is dominated by the constant factors. When the number of examples grows, the kernel matrix $K_{ij} = K(x_i, x_j)$ becomes very large and cannot be stored in memory. Kernel values must be computed on the fly or retrieved from a cache of often accessed values. When the cost of computing each kernel value is relatively high, the kernel cache hit rate becomes a major component of the cost of solving the SVM QP problem [20]. Larger problems must be addressed by using algorithms that access kernel values with very consistent patterns.

Chapter 3

LASVM: An Efficient Online SVM Algorithm

"Learning how to learn is the most important skill in life."

Tony Buzan

The widespread use of computers and advances in storage systems have rapidly increased the amount of digital data that is collected both for scientific and commercial purposes. From a machine learning perspective, having access to more data is desirable since very high dimensional learning systems become theoretically possible when training examples are abundant. On the other hand, processing more data results in a significant increase in the computing cost of algorithms. Even though fast computer processors have vastly enhanced our ability to compute complicated statistical models, the rate at which new data arrives has been outpacing the speed of learning algorithms. Further, if we look at the advances in computer hardware technology in the last decade, hard disks became thousand times bigger but processors became only hundred times faster. Consequently, it has been in-

creasingly important to develop machine learning algorithms that can compensate for this gap between the *production* and the *consumption* of data.

Online learning algorithms are usually associated with problems where the complete training set is not available beforehand. However, their computational properties are very useful for large scale learning. In this chapter, we propose a novel online algorithm, LASVM [2], that outspeeds batch SVM solvers while yielding competitive misclassification rates after a single pass over the training examples. Furthermore, LASVM requires considerably less memory than a regular SVM solver. When combined with an active example selection scheme, LASVM achieves faster training, higher accuracies, and simpler models, using only a fraction of the training example labels.

Support Vector Machines belong to the more general class of learning algorithms called kernel machines. Before introducing the new online SVM algorithm, we first briefly present background on other existing online kernel methods to establish the foundation for the work described in this chapter.

3.1 Kernel Perceptrons

The earliest kernel classifiers [21] were derived from the Perceptron algorithm [22], which represents the decision function (2.15) by maintaining the set S of the indices i of the support vectors. Such online learning algorithms require very little memory because the examples are processed one by one and can be discarded after being examined.

Kernel Perceptron

1) $\mathcal{S} \leftarrow \emptyset, \quad b \leftarrow 0.$

2) Pick a random example (x_t, y_t)

3) Compute $\hat{y}(x_t) = \sum_{i \in \mathcal{S}} \alpha_i \, K(x_t, x_i) + b$

4) If $\quad y_t \, \hat{y}(x_t) \, \leq \, 0 \quad$ then $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{t\}, \quad \alpha_t \leftarrow y_t$

5) Return to step 2.

Iterations such that $y_t \, \hat{y}(x_t) < 0$ are called *mistakes* because they correspond to patterns misclassified by the perceptron decision boundary. The algorithm then modifies the decision boundary by inserting the misclassified pattern into the kernel expansion. When a solution exists, Novikoff's theorem [23] states that the algorithm converges after a finite number of mistakes, or equivalently after inserting a finite number of support vectors. Noisy datasets are more problematic.

3.1.1 Large Margin Kernel Perceptrons

The success of Support Vector Machines has shown that large classification margins were desirable. On the other hand, the Kernel Perceptron makes no attempt to achieve large margins because it happily ignores training examples that are very close to being misclassified.

Many authors have proposed to close the gap with online kernel classifiers by providing larger margins. The Averaged Perceptron [24] decision rule is the majority vote of all the decision rules obtained after each iteration of the Kernel Perceptron algorithm. This choice provides a bound comparable to those offered in support of SVMs. Other algorithms [26, 27, 28, 29] explicitly construct larger margins. These algorithms modify the decision boundary whenever a training example is either misclassified or classified with an insufficient margin. Such examples are

then inserted into the kernel expansion with a suitable coefficient. Unfortunately, this change significantly increases the number of mistakes and therefore the number of support vectors. The increased computational cost and the potential overfitting undermines the positive effects of the increased margin.

3.1.2 Kernel Perceptrons with Removal Step

This is why Crammer et al. [30] suggest an additional step for *removing* support vectors from the kernel expansion (2.15). The Budget Perceptron performs very nicely on relatively clean data sets.

Budget Kernel Perceptron (β, N)

1) $\mathcal{S} \leftarrow \emptyset, \quad b \leftarrow 0.$
2) Pick a random example (x_t, y_t)
3) Compute $\hat{y}(x_t) = \sum_{i \in \mathcal{S}} \alpha_i \, K(x_t, x_i) + b$
4) If $\quad y_t \, \hat{y}(x_t) \; \leq \; \beta \quad$ then,
   - 4a) $\mathcal{S} \leftarrow \mathcal{S} \cup \{t\}, \quad \alpha_t \leftarrow y_t$
   - 4b) If $\; |\mathcal{S}| > N$ then $\mathcal{S} \leftarrow \mathcal{S} - \{ \arg\max_{i \in \mathcal{S}} \; y_i \, (\hat{y}(x_i) - \alpha_i \, K(x_i, x_i)) \}$
5) Return to step 2.

Online kernel classifiers usually experience considerable problems with noisy data sets. Each iteration is likely to cause a mistake because the best achievable misclassification rate for such problems is high. The number of support vectors increases very rapidly and potentially causes overfitting and poor convergence. More sophisticated support vector removal criteria avoid this drawback [31].

3.2 Online Support Vector Machines

This section proposes a novel online algorithm named LASVM that converges to the SVM solution. LASVM relies on the traditional "soft margin" SVM formulation (as described in Chapter 2), handles noisy datasets, and is nicely related to the SMO algorithm. Experimental evidence on multiple datasets indicates that it reliably reaches competitive test error rates after performing a single pass over the training set. It uses less memory and trains significantly faster than state-of-the-art SVM solvers.

3.2.1 Online LASVM

This section presents a novel online SVM algorithm named LASVM. There are two ways to view this algorithm. LASVM is an online kernel classifier sporting a support vector removal step: vectors collected in the current kernel expansion can be removed during the online process. LASVM also is a reorganization of the SMO sequential direction searches and, as such, converges to the solution of the SVM QP problem.

Compared to basic kernel perceptrons [21, 24], the LASVM algorithm features a removal step and gracefully handles noisy data. Compared to kernel perceptrons with removal steps [30, 31], LASVM converges to the known SVM solution. Compared to a traditional SVM solver [12, 13, 14], LASVM brings the computational benefits and the flexibility of online learning algorithms. Experimental evidence indicates that LASVM matches the SVM accuracy after a single sequential pass over the training examples.

This is achieved by alternating two kinds of direction searches named PROCESS and REPROCESS. Each direction search involves a pair of examples. Di-

rection searches of the PROCESS kind involve at least one example that is not a support vector of the current kernel expansion. They potentially can change the coefficient of this example and make it a support vector. Direction searches of the REPROCESS kind involve two examples that already are support vectors in the current kernel expansion. They potentially can zero the coefficient of one or both support vectors and thus remove them from the kernel expansion.

3.2.1.1 Building blocks

The LASVM algorithm maintains three essential pieces of information: the set $\mathcal{S}$ of potential support vector indices, the coefficients α_i of the current kernel expansion, and the partial derivatives g_i defined in (2.25). Variables α_i and g_i contain meaningful values when $i \in \mathcal{S}$ only. The coefficient α_i are assumed to be null if $i \notin \mathcal{S}$. On the other hand, set $\mathcal{S}$ might contain a few indices i such that $\alpha_i = 0$.

LASVM PROCESS(k)

1) Bail out if $k \in \mathcal{S}$.

2) $\alpha_k \leftarrow 0 \ , \quad g_k \leftarrow y_k - \sum_{s \in \mathcal{S}} \alpha_s K_{ks} \ , \quad \mathcal{S} \leftarrow \mathcal{S} \cup \{k\}$

3) If $y_k = +1$ then

$$i \leftarrow k \ , \quad j \leftarrow \arg\min_{s \in S} g_s \ \text{ with } \ \alpha_s > A_s$$

else

$$j \leftarrow k \ , \quad i \leftarrow \arg\max_{s \in S} g_s \ \text{ with } \ \alpha_s < B_s$$

4) Bail out if (i, j) is not a τ-violating pair.

5) $\lambda \leftarrow \min \left\{ \dfrac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}}, \ B_i - \alpha_i, \ \alpha_j - A_j \right\}$

$\alpha_i \leftarrow \alpha_i + \lambda \ , \quad \alpha_j \leftarrow \alpha_j - \lambda$

$g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall \, s \in \mathcal{S}$

The two basic operations of the Online LASVM algorithm correspond to steps 2 and 3 of the SMO algorithm. These two operations differ from each other because they have different ways to select τ-violating pairs.

The first operation, PROCESS, attempts to insert example $k \notin \mathcal{S}$ into the set of current support vectors. In the online setting this can be used to process a new example at time t. It first adds example $k \notin \mathcal{S}$ into $\mathcal{S}$ (step 1-2). Then it searches a second example in $\mathcal{S}$ to find the τ-violating pair with maximal gradient (steps 3-4) and performs a direction search (step 5).

LASVM REPROCESS

1) $i \leftarrow \arg\max_{s \in \mathcal{S}} g_s \quad \text{with} \quad \alpha_s < B_s$

   $j \leftarrow \arg\min_{s \in \mathcal{S}} g_s \quad \text{with} \quad \alpha_s > A_s$

2) Bail out if (i, j) is not a τ-violating pair.

3) $\lambda \leftarrow \min \left\{ \dfrac{g_i - g_j}{K_{ii} + K_{jj} - 2K_{ij}},\ B_i - \alpha_i,\ \alpha_j - A_j \right\}$

   $\alpha_i \leftarrow \alpha_i + \lambda\ , \quad \alpha_j \leftarrow \alpha_j - \lambda$

   $g_s \leftarrow g_s - \lambda(K_{is} - K_{js}) \quad \forall\, s \in \mathcal{S}$

4) $i \leftarrow \arg\max_{s \in \mathcal{S}} g_s \quad \text{with} \quad \alpha_s < B_s$

   $j \leftarrow \arg\min_{s \in \mathcal{S}} g_s \quad \text{with} \quad \alpha_s > A_s$

   For all $s \in \mathcal{S}$ such that $\alpha_s = 0$

   If $y_s = -1$ and $g_s \geq g_i \quad$ then $\quad \mathcal{S} = \mathcal{S} - \{s\}$

   If $y_s = +1$ and $g_s \leq g_j \quad$ then $\quad \mathcal{S} = \mathcal{S} - \{s\}$

5) $b \leftarrow (g_i + g_j)/2\ , \quad \delta \leftarrow g_i - g_j$

The second operation, REPROCESS, removes some elements from $\mathcal{S}$. It first searches the τ-violating pair of elements of $\mathcal{S}$ with maximal gradient (steps 1-2), and performs a direction search (step 3). Then it removes blatant non support

vectors (step 4). Finally it computes two useful quantities: the bias term b of the decision function (2.15) and the gradient δ of the most τ-violating pair in $\mathcal{S}$.

3.2.1.2 Online Iterations of LASVM

After initializing the state variables (step 1), the Online LASVM algorithm alternates PROCESS and REPROCESS a predefined number of times (step 2). Then it simplifies the kernel expansion by running REPROCESS to remove all τ-violating pairs from the kernel expansion (step 3).

LASVM

1) **Initialization**:

   Seed $\mathcal{S}$ with a few examples of each class.

   Set $\boldsymbol{\alpha} \leftarrow \mathbf{0}$ and compute the initial gradient $\boldsymbol{g}$ (equation 2.25)

2) **Online Iterations**:

   Repeat a predefined number of times:

   - Pick an example k_t
   - Run PROCESS(k_t).
   - Run REPROCESS once.

3) **Finishing**:

   Repeat REPROCESS until $\delta \leq \tau$.

LASVM can be used in the online setup where one is given a continuous stream of fresh random examples. The online iterations process fresh training examples as they come. LASVM can also be used as a stochastic optimization algorithm in the offline setup where the complete training set is available before hand. Each iteration randomly picks an example from the training set.

In practice we run the LASVM online iterations in epochs. Each epoch sequentially visits all the randomly shuffled training examples. After a predefined

number P of epochs, we perform the finishing step. A single epoch is consistent with the use of LASVM in the online setup. Multiple epochs are consistent with the use of LASVM as a stochastic optimization algorithm in the offline setup.

3.2.1.3 Convergence of the online iterations

Let us first ignore the finishing step (step 3) and assume that online iterations (step 2) are repeated indefinitely. Suppose that there are remaining τ-violating pairs at iteration T.

a.) If there are τ-violating pairs (i, j) such that $i \in \mathcal{S}$ and $j \in \mathcal{S}$, one of them will be exploited by the next REPROCESS.

b.) Otherwise, if there are τ-violating pairs (i, j) such that $i \in \mathcal{S}$ or $j \in \mathcal{S}$, each subsequent PROCESS has a chance to exploit one of them. The intervening REPROCESS do nothing because they bail out at step 2.

c.) Otherwise, all τ-violating pairs involve indices outside $\mathcal{S}$. Subsequent calls to PROCESS and REPROCESS bail out until we reach a time $t > T$ such that $k_t = i$ and $k_{t+1} = j$ for some τ-violating pair (i, j). The first PROCESS then inserts i into $\mathcal{S}$ and bails out. The following REPROCESS bails out immediately. Finally the second PROCESS locates pair (i, j).

This case is not important in practice. There usually is a support vector $s \in \mathcal{S}$ such that $A_s < \alpha_s < B_s$. We can then write $g_i - g_j = (g_i - g_s) + (g_s - g_j) \leq 2\tau$ and conclude that we already have reached a 2τ-approximate solution.

The LASVM online iterations therefore work like the SMO algorithm. Remaining τ-violating pairs is sooner or later exploited by either PROCESS or REPROCESS. As soon as a τ-approximate solution is reached, the algorithm stops

updating the coefficients $\boldsymbol{\alpha}$.

The finishing step (step 3) is only useful when one limits the number of online iterations. Running LASVM usually consists in performing a predefined number P of epochs and running the finishing step. Each epoch performs n online iterations by sequentially visiting the randomly shuffled training examples. Empirical evidence suggests indeed that a *single epoch* yields a classifier almost as good as the SVM solution.

3.2.1.4 Computational cost of LASVM

Both PROCESS and REPROCESS require a number of operations proportional to the number S of support vectors in set $\mathcal{S}$. Performing P epochs of online iterations requires a number of operations proportional to $n\,P\,\bar{S}$. The average number $\bar{S}$ of support vectors scales no more than linearly with n because each online iteration brings at most one new support vector. The asymptotic cost therefore grows like n^2 at most. The finishing step is similar to running a SMO solver on a SVM problem with only S training examples. We recover here the n^2 to n^3 behavior of standard SVM solvers.

Online algorithms access kernel values with a very specific pattern. Most of the kernel values accessed by PROCESS and REPROCESS involve only support vectors from set $\mathcal{S}$. Only PROCESS on a new example x_{k_t} accesses S fresh kernel values $K(x_{k_t}, x_i)$ for $i \in \mathcal{S}$.

3.2.1.5 Implementation Details

Our LASVM implementation reorders the examples after every PROCESS or REPROCESS to ensure that the current support vectors come first in the reordered list of indices. The kernel cache records truncated rows of the reordered kernel

matrix. SVMLight [20] and LIBSVM [13] also perform such reorderings, but do so rather infrequently [20]. The reordering overhead is acceptable during the online iterations because the computation of fresh kernel values takes much more time.

Reordering examples during the finishing step was more problematic. We eventually deployed an adaptation of the *shrinking* heuristic [20] for the finishing step only. The set $\mathcal{S}$ of support vectors is split into an active set S_a and an inactive set S_i. All support vectors are initially active. The REPROCESS iterations are restricted to the active set S_a and do not perform any reordering. About every 1000 iterations, support vectors that hit the boundaries of the box constraints are either removed from the set $\mathcal{S}$ of support vectors or moved from the active set $\mathcal{S}_a$ to the inactive set $\mathcal{S}_i$. When all τ-violating pairs of the active set are exhausted, the inactive set examples are transferred back into the active set. The process continues as long as the merged set contains τ-violating pairs.

3.2.2 MNIST Experiments

The Online LASVM was first evaluated on the MNIST[1] handwritten digit dataset [32]. Computing kernel values for this dataset is relatively expensive because it involves dot products of 784 gray level pixel values. In the experiments reported below, all algorithms use the same code for computing kernel values. The ten binary classification tasks consist of separating each digit class from the nine remaining classes. All experiments use RBF kernels with $\gamma = 0.005$ and the same training parameters $C = 1000$ and $\tau = 0.001$. Unless indicated otherwise, the kernel cache size is 256MB.

[1] Available at `http://yann.lecun.com/exdb/mnist`

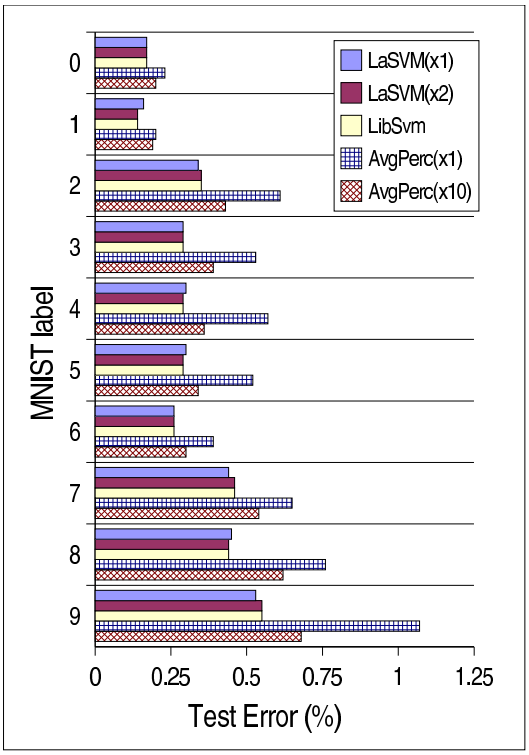


Figure 3.1. *Compared test error rates for the ten MNIST binary classifiers.*

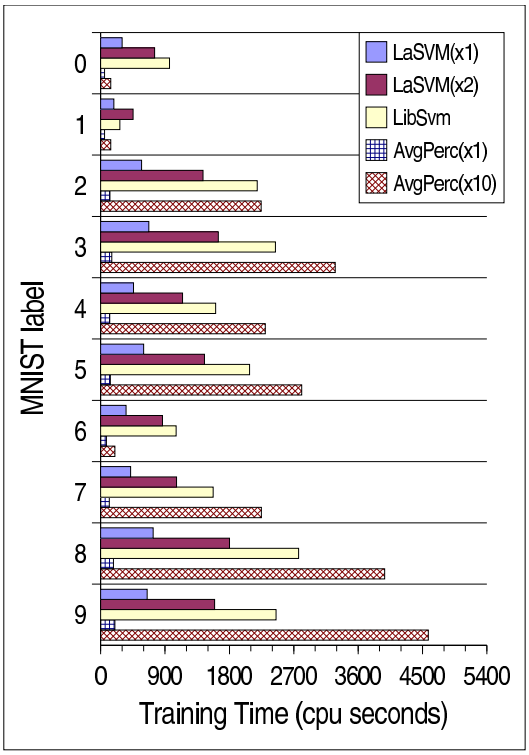


Figure 3.2. *Compared training times for the ten MNIST binary classifiers.*

3.2.2.1 LASVM versus Sequential Minimal Optimization

Baseline results were obtained by running the state-of-the-art SMO solver LIBSVM [13]. The resulting classifier accurately represents the SVM solution.

Two sets of results are reported for LASVM. The LASVM×1 results were obtained by performing a single epoch of online iterations: each training example was processed exactly once during a single sequential sweep over the training set. The LASVM×2 results were obtained by performing two epochs of online iterations.

Figures 3.1 and 3.2 show the resulting test errors and training times. Compared to LIBSVM, LASVM×1 runs about three times faster and yields test error rates very close to the LIBSVM results. Standard paired significance tests indicate that

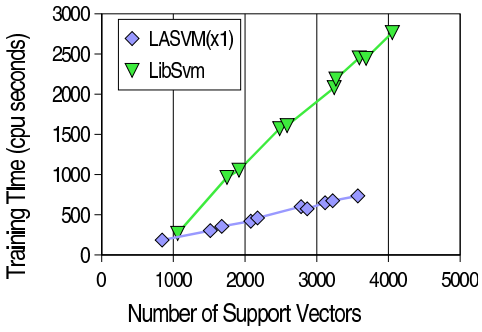


Figure 3.3. *Training time as a function of the number of support vectors.*

Algorithm	Error	Time
LIBSVM	**1.36%**	17400s
LASVM×1	1.42%	**4950s**
LASVM×2	**1.36%**	12210s

Figure 3.4. *Multiclass errors and training times for the MNIST dataset.*

these small differences are not significant. LASVM×2 usually runs faster than LIBSVM and very closely tracks the LIBSVM test errors.

Neither the LASVM×1 or LASVM×2 experiments yield the exact SVM solution. On this dataset, LASVM reaches the exact SVM solution after about five epochs. The first two epochs represent the bulk of the computing time. The remaining epochs run faster when the kernel cache is large enough to hold all the dot products involving support vectors. Yet the overall optimization times are not competitive with those achieved by LIBSVM.

Figure 3.3 shows the training time as a function of the final number of support vectors for the ten binary classification problems. Both LIBSVM and LASVM×1 show a linear dependency. The Online LASVM algorithm seems more efficient overall.

Figure 3.4 shows the multiclass error rates and training times obtained by combining the ten classifiers using the well known 1-versus-rest scheme [33]. LASVM×1 provides almost the same accuracy with much shorter training times. LASVM×2 reproduces the LIBSVM accuracy with slightly shorter training time.

Figure 3.5 shows the resulting number of support vectors. A single epoch of

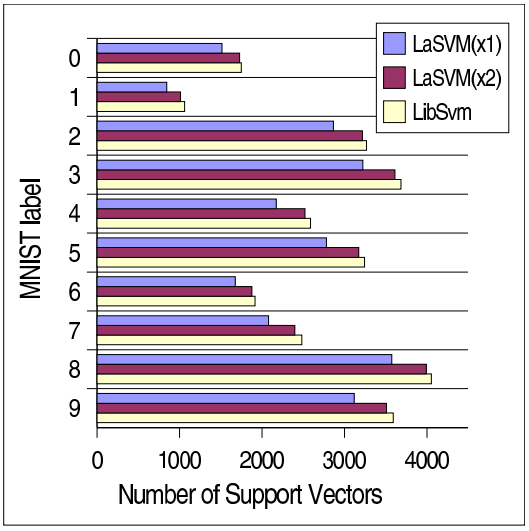


Figure 3.5. *Compared numbers of support vectors for the ten MNIST binary classifiers.*

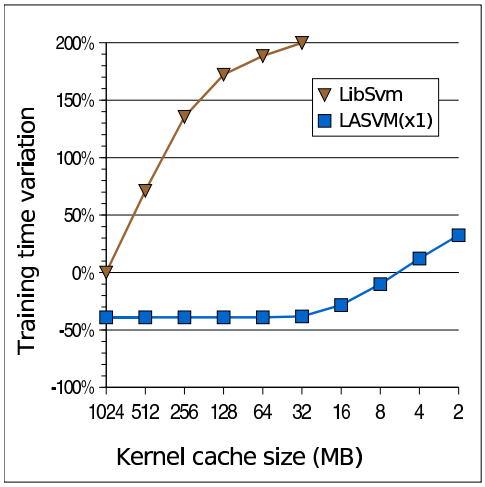


Figure 3.6. *Training time variation as a function of the cache size. Relative changes with respect to the 1GB LIBSVM times are averaged over all ten MNIST classifiers.*

the Online LASVM algorithm gathers most of the support vectors of the SVM solution computed by LIBSVM. The first iterations of the Online LASVM might indeed ignore examples that later become support vectors. Performing a second epoch captures most of the missing support vectors.

3.2.2.2 LASVM versus the Averaged Perceptron

The computational advantage of LASVM relies on its apparent ability to match the SVM accuracies after a single epoch. Therefore it must be compared with algorithms such as the Averaged Perceptron [24] that provably match well known upper bounds on the SVM accuracies. The AVGPERC×1 results in figures 3.1 and 3.2 were obtained after running a single epoch of the Averaged Perceptron. Although the computing times are very good, the corresponding test errors are not com-

petitive with those achieved by either LIBSVM or LASVM. [24] suggest that the Averaged Perceptron approaches the actual SVM accuracies after 10 to 30 epochs. Doing so no longer provides the theoretical guarantees. The AVGPERC×10 results in figures 3.1 and 3.2 were obtained after ten epochs. Test error rates indeed approach the SVM results. The corresponding training times are no longer competitive.

3.2.2.3 Impact of the kernel cache size

These training times stress the importance of the kernel cache size. Figure 3.2 shows how the AVGPERC×10 runs much faster on problems 0, 1, and 6. This is happening because the cache is large enough to accomodate the dot products of all examples with all support vectors. Each repeated iteration of the Average Perceptron requires very few additional kernel evaluations. This is much less likely to happen when the training set size increases. Computing times then increase drastically because repeated kernel evaluations become necessary.

Figure 3.6 compares how the LIBSVM and LASVM×1 training times change with the kernel cache size. The vertical axis reports the relative changes with respect to LIBSVM with one gigabyte of kernel cache. These changes are averaged over the ten MNIST classifiers. The plot shows how LASVM tolerates much smaller caches. On this problem, *LASVM with a 8MB cache runs slightly faster than LIBSVM with a 1024MB cache.*

Useful orders of magnitude can be obtained by evaluating how large the kernel cache must be to avoid the systematic recomputation of dot-products. Let n be the number of examples, S be the number of support vectors, and $R \leq S$ the number of support vectors such that $0 < |\alpha_i| < C$.

- In the case of LIBSVM, the cache must accommodate about $n\,R$ terms: the examples selected for the SMO iterations are usually chosen among the R free support vectors. Each SMO iteration needs n distinct dot-products for each selected example.

- To perform a *single* LASVM epoch, the cache must only accommodate about $S\,R$ terms: since the examples are visited only once, the dot-products computed by a PROCESS operation can only be reutilized by subsequent REPROCESS operations. The examples selected by REPROCESS are usually chosen amont the R free support vectors; for each selected example, REPROCESS needs one distinct dot-product per support vector in set $\mathcal{S}$.

- To perform *multiple* LASVM epochs, the cache must accommodate about $n\,S$ terms: the dot-products computed by processing a particular example are reused when processing the same example again in subsequent epochs. This also applies to multiple Averaged Perceptron epochs.

An efficient single epoch learning algorithm is therefore very desirable when one expects S to be much smaller than n. Unfortunately, this may not be the case when the dataset is noisy. Section 3.2.3 presents results obtained in such less favorable conditions. Section 3.3 then proposes an active learning method to contain the growth of the number of support vectors, and recover the full benefits of the online approach.

3.2.3 Multiple Dataset Experiments

Further experiments were carried out with a collection of standard datasets representing diverse noise conditions, training set sizes, and input dimensionality. Table 3.1 presents these datasets and the parameters used for the experiments.

	Train Size	Test Size	γ	C	Cache	τ	Notes
Waveform[1]	4000	1000	0.05	1	40M	0.001	Artificial data, 21 dims.
Banana[1]	4000	1300	0.5	316	40M	0.001	Artificial data, 2 dims.
Reuters[2]	7700	3299	1	1	40M	0.001	Topic "moneyfx" vs. rest.
USPS[3]	7329	2000	2	1000	40M	0.001	Class "0" vs. rest.
USPS+N[3]	7329	2000	2	10	40M	0.001	10% training label noise.
Adult[3]	32562	16282	0.005	100	40M	0.001	As in [12].
Forest[3] (100k)	100000	50000	1	3	512M	0.001	As in [35].
Forest[3] (521k)	521012	50000	1	3	1250M	0.01	As in [35].

[1] `http://mlg.anu.edu.au/~raetsch/data/index.html`
[2] `http://www.daviddlewis.com/resources/testcollections/reuters21578`
[3] `ftp://ftp.ics.uci.edu/pub/machine-learning-databases`

Table 3.1. *Datasets discussed in section 3.2.3.*

	LIBSVM				LASVM×1			
Dataset	**Error**	**SV**	**KCalc**	**Time**	**Error**	**SV**	**KCalc**	**Time**
Waveform	8.82%	1006	4.2M	3.2s	**8.68%**	**948**	**2.2M**	**2.7s**
Banana	9.96%	873	6.8M	9.9s	9.98%	869	6.7M	10.0s
Reuters	2.76%	1493	11.8M	**24s**	2.76%	1504	**9.2M**	31.4s
USPS	0.41%	236	1.97M	**13.5s**	0.43%	**201**	**1.08M**	15.9s
USPS+N	**0.41%**	2750	63M	305s	0.53%	**2572**	**20M**	**178s**
Adult	14.90%	11327	1760M	1079s	14.94%	11268	**626M**	**809s**
Forest (100k)	**8.03%**	43251	27569M	14598s	8.15%	**41750**	**18939M**	**10310s**
Forest (521k)	4.84%	124782	316750M	159443s	4.83%	122064	**188744M**	**137183s**

Table 3.2. *Comparison of LIBSVM versus LASVM×1: Test error rates (Error), # of support vectors (SV), # of kernel calls (KCalc), and training time (Time). Bold characters indicate significative differences.*

Kernel computation times for these datasets are extremely fast. The data either has low dimensionality or can be represented with sparse vectors. For instance, computing kernel values for two Reuters documents only involves words common to both documents (excluding stop words). The Forest experiments use a kernel implemented with hand optimized assembly code [34].

Table 3.2 compares the solutions returned by LASVM×1 and LIBSVM. The LASVM×1 experiments call the kernel function much less often, but do not always run faster. The fast kernel computation times expose the relative weakness of our kernel cache implementation. The LASVM×1 accuracies are very close to the

Dataset	Relative Variation Error	SV	Time
Waveform	-0%	-0%	+4%
Banana	-79%	-74%	+185%
Reuters	0%	-0%	+3%
USPS	0%	-2%	+0%
USPS+N%	-67%	-33%	+7%
Adult	-13%	-19%	+80%
Forest (100k)	-1%	-24%	+248%
Forest (521k)	-2%	-24%	+84%

Table 3.3. *Relative variations of test error, number of support vectors and training time measured before and after the finishing step.*

LIBSVM accuracies. The number of support vectors is always slightly smaller.

LASVM×1 essentially achieves consistent results over very diverse datasets, after performing one single epoch over the training set only. In this situation, the LASVM PROCESS function gets only once chance to take a particular example into the kernel expansion and potentially make it a support vector. The conservative strategy would be to take all examples and sort them out during the finishing step. The resulting training times would always be worse than LIBSVM's because the finishing step is itself a simplified SMO solver. Therefore LASVM online iterations are able to very quickly discard a large number of examples with a high confidence. This process is not perfect because we can see that the LASVM×1 number of support vectors are smaller than LIBSVM's. Some good support vectors are discarded erroneously.

Table 3.3 reports the relative variations of the test error, number of support vectors, and training time measured before and after the finishing step. The online iterations pretty much select the right support vectors on clean datasets such as "Waveform", "Reuters" or "USPS", and the finishing step does very little. On the other problems the online iterations keep much more examples as potential support

vectors. The finishing step significantly improves the accuracy on noisy datasets such as "Banana", "Adult" or "USPS+N", and drastically increases the computation time on datasets with complicated decision boundaries such as "Banana" or "Forest".

3.2.4 The Collection of Potential Support Vectors

The final step of the REPROCESS operation computes the current value of the kernel expansion bias b and the stopping criterion δ.

$$\begin{aligned} g_{\max} &= \max_{s \in \mathcal{S}} g_s \quad \text{with } \alpha_s < B_s & \qquad b &= \frac{g_{\max} + g_{\min}}{2} \\ g_{\min} &= \min_{s \in \mathcal{S}} g_s \quad \text{with } \alpha_s > A_s & \qquad \delta &= g_{\max} - g_{\min} \end{aligned} \tag{3.1}$$

The quantities $g_{\min}$ and $g_{\max}$ can be interpreted as bounds for the decision threshold b. The quantity δ then represents an uncertainty on the decision threshold b.

The quantity δ also controls how LASVM collects potential support vectors. The definition of PROCESS and the equality (2.25) indicate that PROCESS(k) adds the support vector x_k to the kernel expansion if and only if:

$$y_k \, \hat{y}(x_k) < 1 + \frac{\delta}{2} - \tau \tag{3.2}$$

When $\boldsymbol{\alpha}$ is optimal, the uncertainty δ is zero, and this condition matches the Karush-Kuhn-Tucker condition for support vectors $y_k \, \hat{y}(x_k) \leq 1$.

Intuitively, relation (3.2) describes how PROCESS collects potential support vectors that are compatible with the current uncertainty level δ on the threshold b. Simultaneously, the REPROCESS operations reduce δ and discard the support vectors that are no longer compatible with this reduced uncertainty.

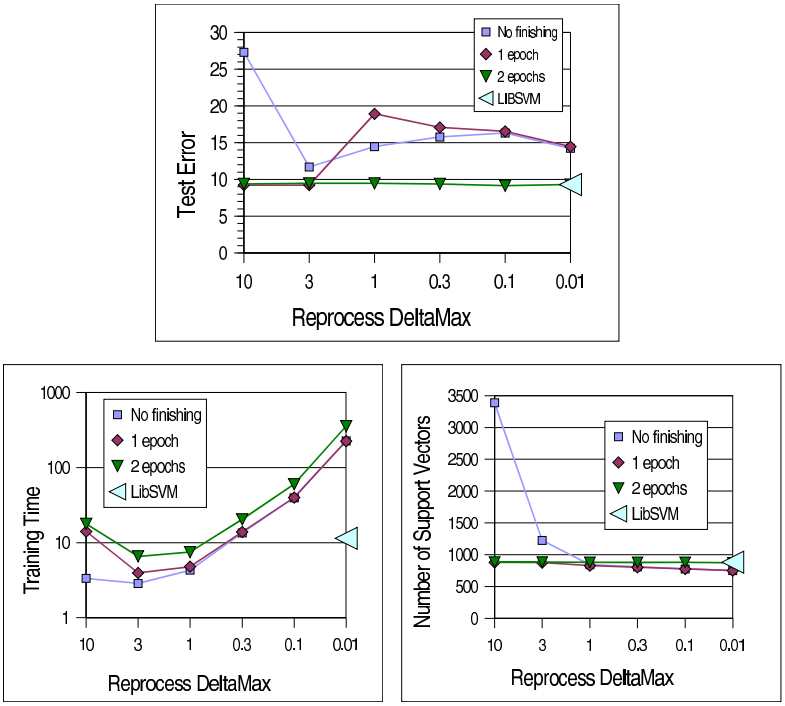


Figure 3.7. *Impact of additional* REPROCESS *measured on "Banana" dataset. During the LASVM online iterations, calls to* REPROCESS *are repeated until* $\delta < \delta_{\max}$.

The online iterations of the LASVM algorithm make equal numbers of PROCESS and REPROCESS for purely heuristic reasons. Nothing guarantees that this is the optimal proportion. The results reported in table 3.3 clearly suggest to investigate this arbitrage more closely.

3.2.4.1 Variations on REPROCESS

Experiments were carried out with a slightly modified LASVM algorithm: instead of performing a single REPROCESS, the modified online iterations repeatedly run REPROCESS until the uncertainty δ becomes smaller than a predefined threshold $\delta_{\max}$.

Figure 3.7 reports comparative results for the "Banana" dataset. Similar results were obtained with other datasets. The three plots report test error rates, training

time, and number of support vectors as a function of $\delta_{\max}$. These measurements were performed after one epoch of online iterations without finishing step, and after one and two epochs followed by the finishing step. The corresponding LIBSVM figures are indicated by large triangles on the right side of the plot.

Regardless of $\delta_{\max}$, the SVM test error rate can be replicated by performing two epochs followed by a finishing step. However, this does not guarantee that the optimal SVM solution has been reached.

Large values of $\delta_{\max}$ essentially correspond to the unmodified LASVM algorithm. Small values of $\delta_{\max}$ considerably increases the computation time because each online iteration calls REPROCESS many times in order to sufficiently reduce δ. Small values of $\delta_{\max}$ also remove the LASVM ability to produce a competitive result after a single epoch followed by a finishing step. The additional optimization effort discards support vectors more aggressively. Additional epochs are necessary to recapture the support vectors that should have been kept.

There clearly is a sweet spot around $\delta_{\max} = 3$ when one epoch of online iterations alone almost match the SVM performance and also makes the finishing step very fast. This sweet spot is difficult to find in general. If $\delta_{\max}$ is a little bit too small, we must make one extra epoch. If $\delta_{\max}$ is a little bit too large, the algorithm behaves like the unmodified LASVM. Short of a deeper understanding of these effects, the unmodified LASVM seems to be a robust compromise.

3.3 Active Selection of Training Examples

The previous section presents LASVM as an Online Learning algorithm or as a Stochastic Optimization algorithm. In both cases, the LASVM online iterations pick random training examples. The current section departs from this framework

and investigates more refined ways to select an informative example for each iteration.

This departure is justified in the offline setup because the complete training set is available beforehand and can be searched for informative examples. It is also justified in the online setup when the continuous stream of fresh training examples is too costly to process, either because the computational requirements are too high, or because it is impractical to label all the potential training examples. Active learning has been theoretically shown to significantly reduce the number of labeled examples needed to find a pattern, both in clean [36] and noisy [37] datasets. In this work, we show that selecting informative examples in online learning setting yields considerable speedups in training as the learner requires less data to reach competitive generalization accuracies and active example selection is insensitive to the artificial label noise. Furthermore, training example selection can be achieved without the knowledge of the training example labels. In fact, excessive reliance on the training example labels can have very detrimental effects.

In active learning, the learner sends a query to select the most informative sample among the training instances. The query function takes center stage in active learning process and determines which instances are presented next to the learner. Cohn et al. [38] and Roy and McCallum [39] propose a query function designed to minimize the error on future test examples. Freund et al. [40] proposed *query by committee* approach wherein the sample that elicits most disagreement among the committee of classifiers is chosen next for querying. Another approach is the uncertainty sampling scheme of Lewis and Catlett [41] where the next sample for querying is the instance on which the learner has the lowest certainty. Tong and Koller [42] suggest a querying approach that aims to reduce the size of the *version space* in each active learning iteration. Similarly, Schohn and Cohn [43],

and Campbell et al. [44] proposed to query the samples close to the classification boundary. The active selection strategy in Section 3.3.2 adopts the approach of minimizing the version space by querying the samples close to the classification boundary. In SVMs, these also correspond to the instances that the learner has the lowest certainty.

3.3.1 Gradient Selection

The most obvious approach consists in selecting an example k such that the PROCESS operation results in a large increase of the dual objective function. This can be approximated by choosing the example which yields the τ-violating pair with the largest gradient. Depending on the class y_k, the PROCESS(k) operation considers pair (k, j) or (i, k) where i and j are the indices of the examples in $\mathcal{S}$ with extreme gradients.

$$i = \arg\max_{s \in \mathcal{S}} g_s \;\; \text{with } \alpha_s < B_s \;, \qquad j = \arg\min_{s \in \mathcal{S}} g_s \;\; \text{with } \alpha_s > A_s$$

The corresponding gradients are $g_k - g_j$ for positive examples and $g_i - g_k$ for negative examples. Using the expression (2.25) of the gradients and the value of b and δ computed during the previous REPROCESS (3.1), we can write:

$$\text{when } y_k{=}{+}1, \quad g_k - g_j = y_k\, g_k - \frac{g_i + g_j}{2} + \frac{g_i - g_j}{2} = 1 + \frac{\delta}{2} - y_k\, \hat{y}(x_k)$$

$$\text{when } y_k{=}{-}1, \quad g_i - g_k = \frac{g_i + g_j}{2} + \frac{g_i - g_j}{2} + y_k\, g_k = 1 + \frac{\delta}{2} - y_k\, \hat{y}(x_k)$$

This expression shows that the *Gradient Selection Criterion* simply suggests to pick the most misclassified example.

$$k_G \quad = \quad \arg\min_{k \notin \mathcal{S}} \quad y_k \, \hat{y}(x_k) \tag{3.3}$$

3.3.2 Active Selection

Always picking the most misclassified example is reasonable when one is very confident of the training example labels. On noisy datasets, this strategy is simply going to pick mislabelled examples or examples that sit on the wrong side of the optimal decision boundary.

When training example labels are unreliable, a conservative approach chooses the example k_A that yields the strongest minimax gradient:

$$k_A \quad = \quad \arg\min_{k \notin \mathcal{S}} \quad \max_{y=\pm 1} \quad y \, \hat{y}(x_k) \quad = \quad \arg\min_{k \notin \mathcal{S}} \quad |\hat{y}(x_k)| \tag{3.4}$$

This *Active Selection Criterion* simply chooses the example that comes closest to the current decision boundary. Such a choice yields a gradient approximatively equal to $1 + \delta/2$ regardless of the true class of the example.

Criterion (3.4) does not depend on the labels y_k. The resulting learning algorithm only uses the labels of examples that have been selected during the previous online iterations. This is related to the *Pool Based Active Learning* paradigm [45].

Early active learning literature, also known as *Experiment Design* [46], contrasts the passive learner, who observes examples (x, y), with the active learner, who constructs queries x and observes their labels y. In this setup, the active learner cannot beat the passive learner because he lacks information about the input pattern distribution [47]. Pool-based active learning algorithms observe the

pattern distribution from a vast pool of unlabelled examples. Instead of constructing queries, they incrementally select unlabelled examples x_k and obtain their labels y_k from an oracle.

Several authors [44, 43, 42] propose incremental active learning algorithms that clearly are related to Active Selection. The initialization consists of obtaining the labels for a small random subset of examples. An SVM is trained using all the labelled examples as a training set. Then one searches the pool for the unlabelled example that comes closest to the SVM decision boundary, one obtains the label of this example, retrains the SVM and reiterates the process.

3.3.3 Small Pool Active Learning

Both criteria (3.3) and (3.4) suggest a search through all the training examples. This is impossible in the online setup and potentially expensive in the offline setup.

It is however possible to locate an approximate optimum by simply examining a small constant number of randomly chosen examples. Small pool active learning first samples M random training examples and selects the best one among these M examples. With probability $1 - \eta^M$, the value of the criterion for this example exceeds the η-quantile of the criterion for all training examples [33, theorem 6.33] regardless of the size of the training set. Section 4.2.1 provides further details on this sampling scheme.

Small pool active learning has been used in the offline setup to accelerate various machine learning algorithms [48, 49, 50]. In the online setup, randomized search is the only practical way to select training examples. For instance, here is a modification of the basic LASVM algorithm to select examples using the Active Selection Criterion with small pools:

LASVM + Small Pool Active Learning

1) **Initialization**:

   Seed $\mathcal{S}$ with a few examples of each class.

   Set $\boldsymbol{\alpha} \leftarrow \mathbf{0}$ and $\boldsymbol{g} \leftarrow \mathbf{0}$.

2) **Online Iterations**:

   Repeat a predefined number of times:

   - Pick M random examples $s_1 \ldots s_M$.
   - $k_t \leftarrow \underset{i=1\ldots M}{\arg\min} \mid \hat{y}(x_{s_i}) \mid$
   - Run PROCESS(k_t).
   - Run REPROCESS once.

3) **Finishing**:

   Repeat REPROCESS until $\delta \leq \tau$.

Each online iteration of the above algorithm is about M times more computationally expensive than an online iteration of the basic LASVM algorithm. Indeed one must compute the kernel expansion (2.15) for M fresh examples instead of a single one (2.25). This cost can be reduced by heuristic techniques for adapting M to the current conditions. For instance, we present experimental results where one stops collecting new examples as soon as $\mathcal{M}$ contains five examples such that $\mid \hat{y}(x_s) \mid < 1 + \delta/2$.

3.3.4 Example Selection for Online SVMs

This section experimentally compares the LASVM algorithm using different example selection methods. Four different algorithms are compared:

- RANDOM example selection strategy randomly picks the next training example among those that have not yet been PROCESSed. This is equivalent

to the plain LASVM algorithm discussed in section 3.2.1.

- GRADIENT example selection consists in sampling 50 random training examples among those that have not yet been PROCESSed. The sampled example with the smallest $y_k\,\hat{y}(x_k)$ is then selected.

- ACTIVE example selection consists in sampling 50 random training examples among those that have not yet been PROCESSed. The sampled example with the smallest $|\hat{y}(x_k)|$ is then selected.

- AUTOACTIVE example selection attempts to adaptively select the sampling size. Sampling stops as soon as 5 examples are within distance $1+\delta/2$ of the decision boundary. The maximum sample size is 100 examples. The sampled example with the smallest $|\hat{y}(x_k)|$ is then selected.

3.3.4.1 Adult Dataset

We first report experiments performed on the Adult dataset. This dataset provides a good indication of the relative performance of the Gradient and Active selection criteria under noisy conditions.

Reliable results were obtained by averaging experimental results measured for 65 random splits of the full dataset into training and test sets. Paired tests indicate that test error differences of 0.25% on a single run are statistically significant at the 95% level. We conservatively estimate that average error differences of 0.05% are meaningful.

Figure 3.8 reports the average error rate measured on the test set as a function of the number of online iterations (left plot) and of the average computing time (right plot). Regardless of the training example selection method, all reported results were measured after performing the LASVM finishing step. More specifically,

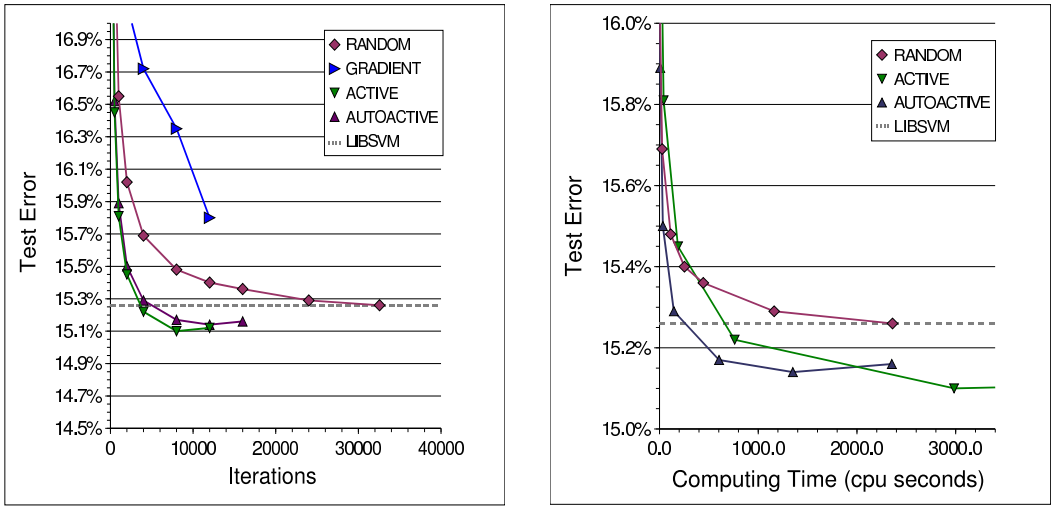


Figure 3.8. *Comparing example selection criteria on the Adult dataset. Measurements were performed on 65 runs using randomly selected training sets. The graphs show the error measured on the remaining testing examples as a function of the number of iterations and the computing time. The dashed line represents the LIBSVM test error under the same conditions.*

we run a predefined number of online iterations, save the LASVM state, perform the finishing step, measure error rates and number of support vectors, and restore the saved LASVM state before proceeding with more online iterations. Computing time includes the duration of the online iterations and the duration of the finishing step.

The GRADIENT example selection criterion performs very poorly on this noisy dataset. A detailed analysis shows that most of the selected examples become support vectors with coefficient reaching the upper bound C. The ACTIVE and AUTOACTIVE criteria both reach smaller test error rates than those achieved by the SVM solution computed by LIBSVM. The error rates then seem to increase towards the error rate of the SVM solution (left plot). We believe indeed that continued iterations of the algorithm eventually yield the SVM solution.

Figure 3.9 relates error rates and numbers of support vectors. The RANDOM

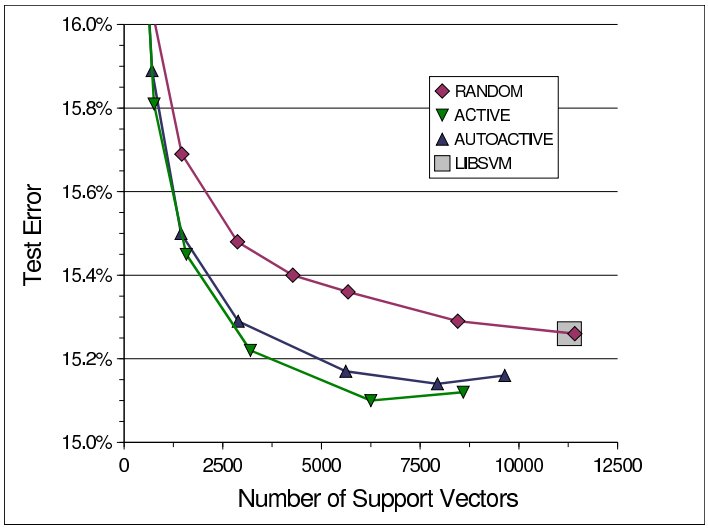


Figure 3.9. *Comparing example selection criteria on the Adult dataset. Test error as a function of the number of support vectors.*

LASVM algorithm performs as expected: a single pass over all training examples replicates the accuracy and the number of support vectors of the LIBSVM solution. Both the ACTIVE and AUTOACTIVE criteria yield kernel classifiers with the same accuracy and much less support vectors. For instance, the AUTOACTIVE LASVM algorithm reaches the accuracy of the LIBSVM solution using 2500 support vectors instead of 11278. Figure 3.8 (right plot) shows that this result is achieved after 150 seconds only. This is about one fifteenth of the time needed to perform a full RANDOM LASVM epoch.

Both the ACTIVE LASVM and AUTOACTIVE LASVM algorithms exceed the LIBSVM accuracy after a few iterations only. This is surprising because these algorithms only use the training labels of the few selected examples. They both outperform the LIBSVM solution by using only a small subset of the available training labels.

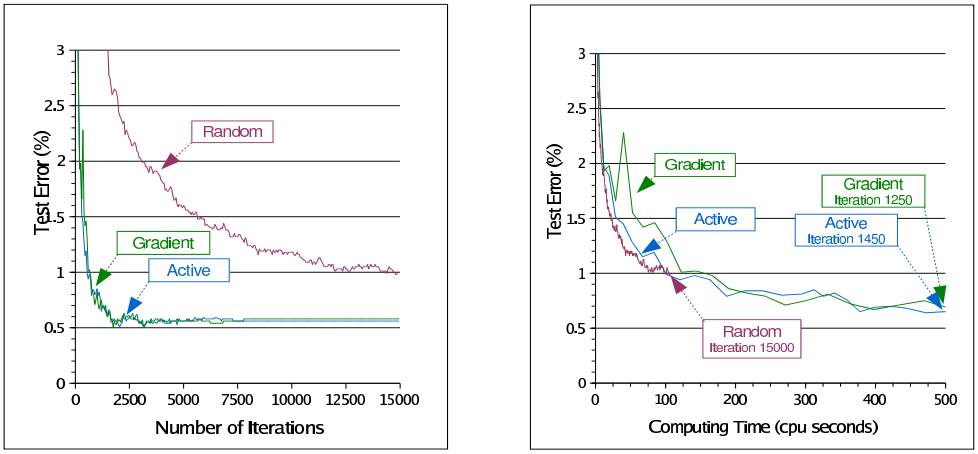


Figure 3.10. *Comparing example selection criteria on the MNIST dataset, recognizing digit "8" against all other classes. Gradient selection and Active selection perform similarly on this relatively noiseless task.*

3.3.4.2 MNIST Dataset

The comparatively clean MNIST dataset provides a good opportunity to verify the behavior of the various example selection criteria on a problem with a much lower error rate.

Figure 3.10 compares the performance of the RANDOM, GRADIENT and ACTIVE criteria on the classification of digit "8" versus all other digits. The curves are averaged on 5 runs using different random seeds. All runs use the standard MNIST training and test sets. Both the GRADIENT and ACTIVE criteria perform similarly on this relatively clean dataset. They require about as much computing time as RANDOM example selection to achieve a similar test error.

Adding ten percent label noise on the MNIST training data provides additional insight regarding the relation between noisy data and example selection criteria. Label noise was not applied to the testing set because the resulting measurement can be readily compared to test errors achieved by training SVMs without label

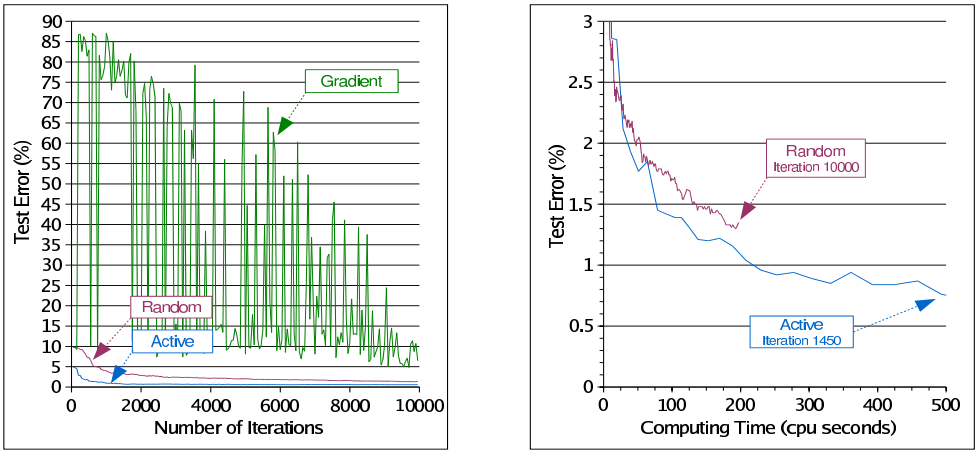


Figure 3.11. *Comparing example selection criteria on the MNIST dataset with 10% label noise on the training examples.*

noise. The expected test errors under similar label noise conditions can be derived from the test errors measured without label noise. Figure 3.11 shows the test errors achieved when 10% label noise is added to the training examples. The GRADIENT selection criterion causes a very chaotic convergence because it keeps selecting mislabelled training examples. The ACTIVE selection criterion is obviously undisturbed by the label noise.

Figure 3.12 summarizes error rates and number of support vectors for all noise conditions. In the presence of label noise on the training data, LIBSVM yields a slightly higher test error rate, and a much larger number of support vectors. The RANDOM LASVM algorithm replicates the LIBSVM results after one epoch. Regardless of the noise conditions, the ACTIVE LASVM algorithm reaches the accuracy and the number of support vectors of the LIBSVM solution obtained with clean training data. Although we have not been able to observe it on this dataset, we expect that, after a very long time, the ACTIVE curve for the noisy training set converges to the accuracy and the number of support vectors achieved of the LIBSVM solution obtained for the noisy training data.

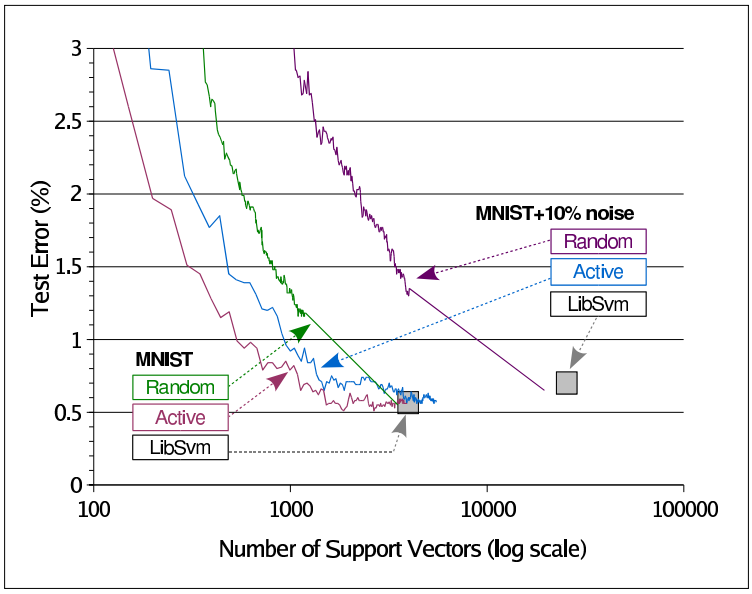


Figure 3.12. *Comparing example selection criteria on the MNIST dataset. Active example selection is insensitive to the artificial label noise.*

3.3.5 Active Learning in Online and Batch Settings

The ACTIVE LASVM algorithm implements two dramatic speedups with respect to existing active learning algorithms such as [44, 43, 42]. First it chooses a query by sampling a small number of random examples instead of scanning all unlabelled examples. Second, it uses a single LASVM iteration after each query instead of fully retraining the SVM.

Figure 3.13 reports experiments performed on the Reuters and USPS datasets presented in table 3.1. The RETRAIN ACTIVE 50 and RETRAIN ACTIVE ALL select a query from 50 or all unlabeled examples respectively, and then retrain the SVM. The SVM solver was initialized with the solution from the previous iteration. The LASVM ACTIVE 50 and LASVM ACTIVE ALL do not retrain the SVM, but instead make a single LASVM iteration for each new labeled example.

All the active learning methods performed approximately the same, and were superior to random selection. Using LASVM iterations instead of retraining causes

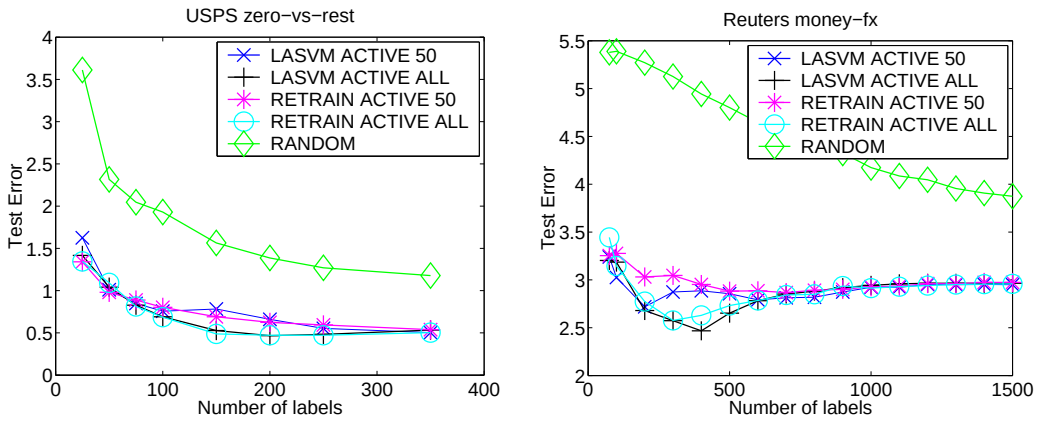


Figure 3.13. *Comparing active learning methods on USPS and Reuters datasets. Results are averaged on 10 random choices of training and test sets. Using LASVM iterations instead of retraining causes no loss of accuracy. Sampling $M = 50$ examples instead of searching all examples only causes a minor loss of accuracy when the number of labeled examples is very small.*

no loss of accuracy. Sampling $M = 50$ examples instead of searching all examples only causes a minor loss of accuracy when the number of labeled examples is very small. Yet the speedups are very significant: for 500 queried labels on the Reuters dataset, the RETRAIN ACTIVE ALL, LASVM ACTIVE ALL, and LASVM ACTIVE 50 algorithms took 917 seconds, 99 seconds, and 9.6 seconds respectively.

3.4 Name Disambiguation in CiteSeer Repository Using LASVM and Active Learning

We assessed the effectiveness of LASVM and active learning in real world problems by targeting the name disambiguation problem in CiteSeer's repository of scientific publications. Name disambiguation can occur when one is seeking a list of publications of an author who has used different name variations and when there are multiple other authors with the same name. We developed an efficient name

disambiguation framework [51] that integrates supervised and unsupervised methods to provide a scalable and efficient solution. We give a brief overview of our methodology, followed by the experimental results on the CiteSeer collection.

Given a research paper, each author appearance in that paper is associated with a metadata record, which has a set of attributes such as addresses, affiliations, email and urls. For each pair of records, we compute a similarity vector based on the "distance" of these records. We use different similarity predicates depending on the nature of the attributes. For instance, we use the edit distance for emails and urls, token-based Jaccard similarity for addresses and affilitations, hybrid similarity Soft-TFIDF [52] for name variations. We then use LASVM to train a model that can predict whether a pair of records identify the same person (i.e. coreferent). For each record pair in the test set, we use the classification confidence of SVM as a pairwise point distance metric, which we then construct clusters of records based on multiple pairwise distances using DBSCAN [53]. The next section presents empirical evaluation of coreferent identification using LASVM with active learning. For further details and experimental results on DBSCAN based clustering, please refer to [51].

3.4.1 Experiments on SVM Based Distance Function

We empirically study the efficiency and effectiveness of our proposed method for name disambiguation by first testing the performance of the supervised distance function, and then the performance of the entire framework.

Using the CiteSeer metadata dataset, ten most ambiguous names are sampled from the entire dataset as listed in Table 3.4 (where **R** is the number of unique records and **A** the number of unique authors). These names are in parallel with

the names used in [54, 55], and are geographically diverse to cover names of different origins. 3,355 papers are manually labeled yielding 490 authors. For those ambiguous author names from different papers, we meticulously went through the original papers, homepages, CVs, etc; and for some, we sent emails to confirm their authorship. These sampled datasets represent the worst case scenario.

We measure the performance of the SVM based distance function by its capability to correctly classify a similarity vector as coreferent or not. We also measure the prediction time of the SVM model for distance calculation as it's crucial to the scalability of the entire system.

We select datasets with ID number 4, 5 and 9 in Table 3.4 as a three-fold training dataset, consisting of 81,073 pieces of pairwise coreference training samples drawn from candidate classes. We conduct grid search to obtain the optimal parameters C(=1) and γ(=0.05) for the RBF kernel in SVM. Our first goal is to obtain a simpler model to efficiently calculate the distances between record pairs. We believe that we can achieve this simpler model by using only the most informative training samples. We conduct three-fold cross validation with active learning

Table 3.4. Author datasets (R=#records, A=#authors)

ID	Dataset	R	A
1	A. Gupta	506	44
2	A. Kumar	143	36
3	C. Chen	536	103
4	D. Johnson	350	41
5	J. Anderson	327	43
6	J. Robinson	115	30
7	J. Smith	743	86
8	K. Tanaka	53	20
9	M. Jones	352	53
10	M. Miller	230	34
	Total	3,355	490

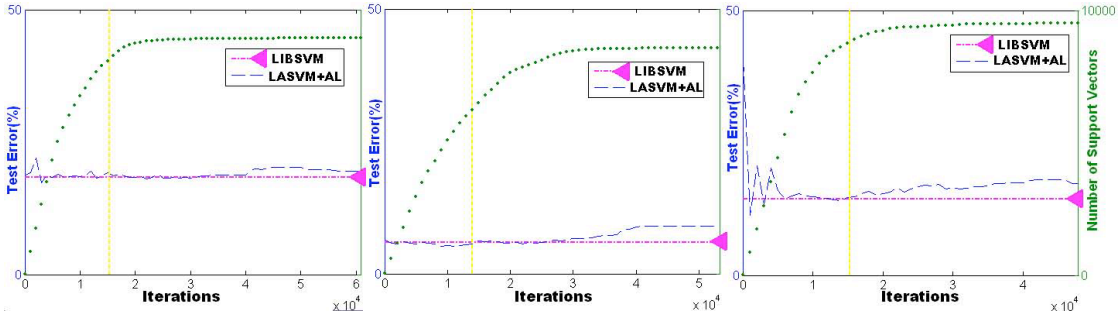


Figure 3.14. Cross-validation on three-fold training datasets (from left to right: train[4,5] test[9]; train[4,9] test[5]; train[5,9] test[4]). The optimal iteration number for early stopping is shown with a yellow horizontal line. The LIBSVM test error is indicated by a pink triangle and the number of iterations refers to LASVM only.

setup on the entire training dataset. As we see in Fig. 3.14, in active learning setups, after using certain number of labeled training data, the number of support vectors saturates and the test error becomes more stable. We observe that adding more training data after this point does not make a remarkable change in the model. The reason is that with active learning method, the most informative samples are already included in the model, thus the remaining samples do not provide extra information. Therefore, we determine an early stopping point for training the model from cross validation results (shown in Fig. 3.14). We first select an interval of iteration number from 12,310 to 14,100, where the average cross validation error is stably minimized. Then we fix the iteration number to 14,100, where the number of support vectors is closest to saturation.

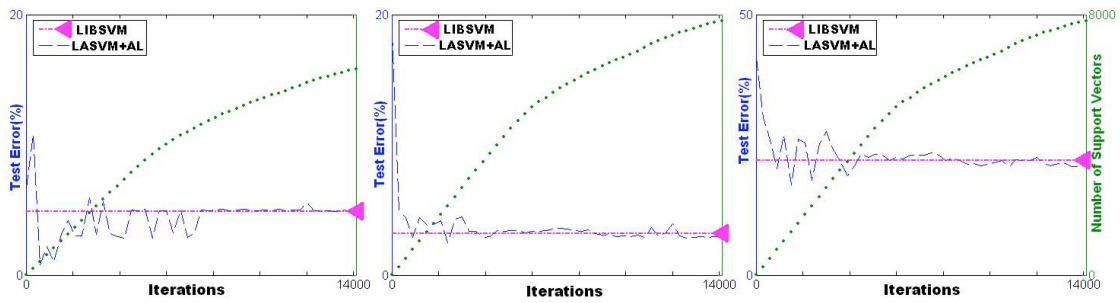


Figure 3.15. Test errors in different iterations for test datasets 3, 6 and 10.

Table 3.5. SVM Models Testing Results

ID	**Error(%)**		**Prediction Time(sec.)**[2]	
	LIBSVM	LASVM(%chg.)	LIBSVM	LASVM(%chg.)
1	19.34	**17.989**(-7.00%)	137.3	**109.3(-20.4%)**
2	6.491	**6.149**(-5.26%)	6.3	**5.1(-19.0%)**
3	**4.882**	4.885(+0.07%)	118.8	**94.2(-20.7%)**
6	2.814	**2.335**(-17.0%)	5.3	**4.1(-22.6%)**
7	9.721	**9.168**(-5.69%)	215.6	**170.2(-21.1%)**
8	11.00	**10.513**(-4.45%)	1.1	**0.8(-27.3%)**
10	21.31	**18.35(-13.9%)**	25.3	**19.6(-22.5%)**
Avg	11.218	**9.913(-7.60%)**	72.8	**57.6(-23.5%)**

Our LASVM model is trained on the entire training dataset, stopping the training process at this iteration number to get a simpler model. We also train LIBSVM on this dataset for comparison purposes. Table 3.5 shows the test error and prediction time of LASVM, compared to classical SVM, for the seven test datasets. Our model demonstrates an average of 23.5% decrease in the prediction time, which can be mostly explained by the decrease in the number of support vectors from 9,822 to 7,809. This simpler model also achieves 7.6% decrease in test error, implying a more accurate distance function. Fig. 3.15 shows how test error changes with the number of iterations for test datasets 3, 6 and 10. These figures show that with early stopping, we can achieve the same or even lower test error rates by using only the most informative portion of all training samples.

To test the efficiency of the name disambiguation framework which comprises of LASVM based distance function with active sample selection followed by DBSCAN clustering, we disambiguate the entire CiteSeer metadata datasets in 3,880 minutes, yielding 418,809 unique authors from more than 700,000 scientific publications. The most prolific authors in the CiteSeer digital library are reported in [51].

3.5 Discussions on LASVM

3.5.1 Practical Significance

This work started because we observed that the dataset sizes are quickly outgrowing the computing power of our calculators. One possible avenue consists in harnessing the computing power of multiple computers [34]. Another line of research consists in seeking learning algorithms with low complexity.

When we have access to an abundant source of training examples, the simple way to reduce the complexity of a learning algorithm consists of picking a random subset of training examples and running a regular training algorithm on this subset. Unfortunately this approach renounces the more accurate models that the large training set could afford. This is why we say, by reference to statistical efficiency, that an *efficient* learning algorithm should at least pay a brief look at every training example.

The LASVM algorithm is very attractive because it yields competitive results after a single epoch. This is very important in practice because modern data storage devices are most effective when the data is accessed sequentially.

Active Selection of the LASVM training examples brings two additional benefits for practical applications: (a) it achieves equivalent performances with significantly less support vectors, and (b) the search for informative examples is an obviously parallel task.

3.5.2 Informative Examples and Support Vectors

By suggesting that all examples should not be given equal attention, we first state that all training examples are not equally informative. This question has been

asked and answered in various contexts [46, 45, 56]. We also ask whether these differences can be exploited to reduce the computational requirements of learning algorithms. Our work answers this question by proposing algorithms that exploit these differences and achieve very competitive performances.

Kernel classifiers in general distinguish the few training examples named support vectors. Kernel classifier algorithms usually maintain an active set of potential support vectors and work by iterations. Their computing requirements are readily associated with the training examples that belong to the active set. Adding a training example to the active set increases the computing time associated with each subsequent iteration because they will require additional kernel computations involving this new support vector. Removing a training example from the active set reduces the cost of each subsequent iteration. However it is unclear how such changes affect the number of subsequent iterations needed to reach a satisfactory performance level.

Online kernel algorithms, such as the kernel perceptrons usually produce different classifiers when given different sequences of training examples. Section 3.2 proposes an online kernel algorithm that converges to the SVM solution after many epochs. The final set of support vectors is intrinsically defined by the SVM QP problem, regardless of the path followed by the online learning process. Intrinsic support vectors provide a benchmark to evaluate the impact of changes in the active set of current support vectors. Augmenting the active set with an example that is not an intrinsic support vector moderately increases the cost of each iteration without clear benefits. Discarding an example that is an intrinsic support vector incurs a much higher cost. Additional iterations will be necessary to recapture the missing support vector. Empirical evidence is presented in section 3.2.4.

Nothing guarantees however that the most informative examples are the sup-

port vectors of the SVM solution. Bakir et al. [57] interpret Steinwart's theorem [19] as an indication that the number of SVM support vectors is asymptotically driven by the examples located on the wrong side of the optimal decision boundary. Although such outliers might play a useful role in the construction of a decision boundary, it seems unwise to give them the bulk of the available computing time. Section 3.3 adds explicit example selection criteria to LASVM. The Gradient Selection Criterion selects the example most likely to cause a large increase of the SVM objective function. Experiments show that it prefers outliers over honest examples. The Active Selection Criterion bypasses the problem by choosing examples without regard to their labels. Experiments show that it leads to competitive test error rates after a shorter time, with less support vectors, and using only the labels of a small fraction of the examples.

3.6 Remarks

This work explores various ways to speedup kernel classifiers by asking which examples deserve more computing time. We have proposed a novel online algorithm that converges to the SVM solution. LASVM reliably reaches competitive accuracies after performing a single pass over the training examples, outspeeding state-of-the-art SVM solvers. We have then shown how active example selection can yield faster training, higher accuracies and simpler models using only a fraction of the training examples labels.

Chapter 4

Active Learning in Imbalanced Data Classification

"Life is like riding a bicycle. To keep your balance, you must keep moving."

Albert Einstein

In classification problems, a training dataset is called imbalanced if at least one of the classes are represented by significantly less number of instances (i.e. observations, examples, cases) than the others. Real world applications often face this problem because naturally normal examples which constitute the majority class are generally abundant; on the other hand the examples of interest are generally rare and form the minority class. Another reason for class imbalance problem is the limitations (e.g., cost, difficulty or privacy) on collecting instances of some classes. Examples of applications which may have class imbalance problem include, but are not limited to, predicting pre-term births [58], identifying fraudulent credit card transactions [59], text categorization [60], classification of protein databases [61] and detecting certain objects from satellite images [62]. Despite that they

are difficult to identify, rare instances generally constitute the target concept in classification tasks. However, in imbalanced data classification, the class boundary learned by standard machine learning algorithms can be severely skewed toward the target class. As a result, the false-negative rate can be excessively high.

In classification tasks, it is generally more important to correctly classify the minority class instances. In real-world applications, mispredicting a rare event can result in more serious consequences than mispredicting a common event. For example in the case of cancerous cell detection, misclassifying non-cancerous cells leads to additional clinical testing but misclassifying cancerous cells leads to very serious health risks. Similar problem might occur in detection of a threatening surveillance event from video streams, where misclassifying a normal event may only result in increased security but misclassifying a life threatening event may lead to disastrous consequences. However in classification problems with imbalanced data, the minority class examples are more likely to be misclassified than the majority class examples. Due to their design principles, most of the machine learning algorithms optimize the overall classification accuracy hence sacrifice the prediction performance on the minority classes. We propose an efficient active learning framework [3] which has high prediction performance to overcome this serious data mining problem.

In addition to the naturally occurring class imbalance problem, the imbalanced data situation may also occur in one-against-rest schema in multiclass classification. Assuming there are N different classes, one of the simplest multiclass classification schemes built on top of binary classifiers is to train N different binary classifiers. Each classifier is trained to distinguish the examples in a single class from the examples in all remaining classes. When it is desired to classify a new example, the N classifiers are run, and the classifier which has the highest classifi-

cation confidence is chosen. Therefore, even though the training data is balanced, issues related to the class imbalance problem can frequently surface.

This chapter presents an alternative to the existing methods: using active learning strategy to deal with the class imbalance problem [3]. Active learning has been pronounced by some researchers [63, 64] as a sampling method but no systematic study has been done to show that it works well with imbalanced data. We demonstrate that by selecting informative instances for training, active learning can indeed be a useful technique to address the class imbalance problem. We constrain our discussion to the standard two-class classification problem with Support Vector Machines. In the rest of the chapter, we refer to the minority and majority classes as "positive" and "negative" respectively.

We propose an efficient SVM based active learning selection strategy which queries small pool of data at each iterative step instead of querying the entire dataset. The proposed method brings the advantage of efficient querying in search of the most informative instances, thus enabling active learning strategy to be applied to large datasets without high computational costs. Rather than using a traditional batch SVM, we use the LASVM algorithm presented in Chapter 3, which suits better to the nature of active learning due to its incremental learning steps. We present that active learning's querying strategy yields a balanced training set in the early stages of the learning without any requirement of preprocessing of the data. Major research direction in recent literature to overcome the class imbalance problem is to resample the original training dataset to create more balanced classes. This is done either by oversampling the minority class and/or undersampling the majority class until the classes are approximately equally represented. Our empirical results show that active learning can be a more efficient alternative to resampling methods in creating balanced training set for the learner.

AL does not risk losing information as in undersampling and does not bring an extra burden of data as in oversampling. With early stopping, active learning can achieve faster and scalable solution without sacrificing prediction performance.

4.1 Related Work on Class Imbalance Problem

Recent research on class imbalance problem has focused on several major groups of techniques. One is to assign distinct costs to the classification errors [65, 66]. In this method, the misclassification penalty for the positive class is assigned a higher value than that of the negative class. This method requires tuning to come up with good penalty parameters for the misclassified examples. The second is to resample the original training dataset, either by over-sampling the minority class and/or under-sampling the majority class until the classes are approximately equally represented [67, 68, 69, 70]. Both resampling methods introduce additional computational costs of data preprocessing and oversampling can be overwhelming in the case of very large scale training data. Undersampling has been proposed as a good means of increasing the sensitivity of a classifier. However this method may discard potentially useful data that could be important for the learning process therefore significant decrease in the prediction performance may be observed. Discarding the redundant examples in undersampling has been discussed in [71] but since it is an adaptive method for ensemble learning and does not involve an external preprocessing step it can not be applied to other types of algorithms. Oversampling has been proposed to create synthetic positive instances from the existing positive samples to increase the representation of the class. Nevertheless, oversampling may suffer from overfitting and due to the increase in the number of samples, the training time of the learning process gets longer. If a complex oversampling

method is used, it also suffers from high computational costs during preprocessing data. In addition to those, oversampling methods demand more memory space for the storage of newly created instances and the data structures based on the learning algorithm (i.e., extended kernel matrix in kernel classification algorithms). Deciding on the oversampling and undersampling rate is also another issue of those methods.

Another technique suggested for class imbalance problem is to use a recognition-based, instead of discrimination-based inductive learning [72, 73]. These methods attempt to measure the amount of similarity between a query object and the target class, where classification is accomplished by imposing a threshold on the similarity measure. The major drawback of those methods is the need for tuning the similarity threshold of which the success of the method mostly relies on. On the other hand, discrimination-based learning algorithms have been proved to give better prediction performance in most domains.

Akbani et al. [74] investigate the behavior of Support Vector Machines (SVM) with imbalanced data. They applied SMOTE algorithm [67] to oversample the data and trained SVM with different error costs. SMOTE is an oversampling approach in which the minority class is oversampled by creating synthetic examples rather than with replacement. The k nearest positive neighbors of all positive instances are identified and synthetic positive examples are created and placed randomly along the line segments joining the k minority class nearest neighbors. Preprocessing the data with SMOTE may lead to improved prediction performance at the classifiers, however it also brings more computational cost to the system for preprocessing and yet the increased number of training data makes the SVM training very costly since the training time at SVMs scales quadratically with the number of training instances. In order to cope with today's tremendously growing

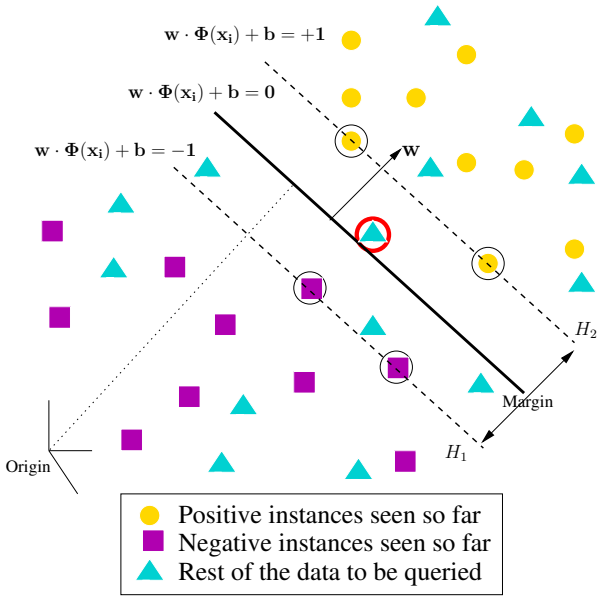


Figure 4.1. Active Learning with SVM (separable case). The most informative sample among the unseen training samples is the one (in bold circle) closest to the hyperplane (solid line). The circled samples on the dashed lines are support vectors.

dataset sizes, we believe that there is a need for more computationally efficient and scalable algorithms. We show that such a solution can be achieved by using active learning strategy.

4.2 Methodology

Active learning is mostly regarded as a technique that addresses the unlabeled training instance problem. The learner has access to a vast pool of unlabeled examples, and it tries to make a clever choice to select the most informative example to obtain its label. However, in the cases where all the labels are available beforehand, active learning can still be leveraged to obtain the informative instances through the training sets [43, 2, 51]. In SVMs, *informativeness* of an instance is synonymous with its distance to the hyperplane. The farther an instance is to the hyperplane, the more the learner is confident about its true class label, hence it

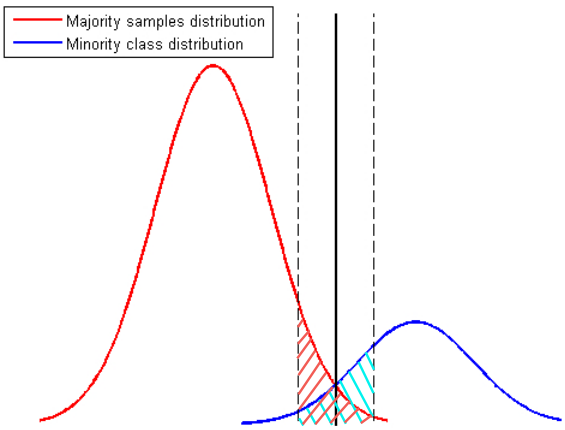


Figure 4.2. Data within the margin is less imbalanced than the entire data.

does not bring much (or any) information to the system. On the other hand, the instances close to the hyperplane are informative for learning. SVM based active learning can pick up the informative instances by checking their distances to the hyperplane. The closest instances to the hyperplane are considered to be the most informative instances.

The strategy of selecting instances within the margin addresses the imbalanced dataset classification very well. Suppose that the class distributions of an imbalanced dataset is given in Figure 4.2. The shaded region corresponds to the class distribution of the data within the margin. As it can be observed, the imbalance ratio of the classes within the margin is much smaller than the class imbalance ratio of the entire dataset. Any selection strategy which focuses on the instances in the margin most likely ends up with a more balanced class distribution than that of the entire dataset. Our empirical findings with various type of real-world data confirm that the imbalance ratios of the classes within the margin in real-world data are generally much lower than that of the entire data as shown in Figure 4.2.

We present the working principles of the efficient active learning algorithm in

Section 4.2.1. We continue with explaining the advantage of using online SVMs with the active sample selection in Section 4.2.2. In Section 4.2.3, we then describe an early stopping heuristics for active learning.

4.2.1 Active Learning

Remember from equation 2.7 that only the support vectors have an effect on the SVM solution. This means that if SVM is retrained with a new set of data which only consist of those support vectors, the learner will end up finding the same hyperplane. This fact leads us to the idea that not all the instances are equally important in the training sets. Then the question becomes how to select the most informative examples in the datasets. We will focus on a form of selection strategy called SVM based active learning. In SVMs, the most informative instance is believed to be the closest instance to the hyperplane since it divides the *version space* into two equal parts. The aim is to reduce the version space as fast as possible to reach the solution faster in order to avoid certain *costs* associated with the problem. For the possibility of a non-symmetric version space, there are more complex selection methods suggested by [42], but it has been observed that the advantage of those are not significant when compared to their high computational costs.

Active Learning with Small Pools: The basic working principle of SVM active learning is: *i*) learn an SVM on the existing training data, *ii*) select the closest instance to the hyperplane, and *iii*) add the new selected instance to the training set and train again. In classical active learning [42], the search for the most informative instance is performed over the entire dataset. Note that, each iteration of active learning involves the recomputation of each training example's

distance to the new hyperplane. Therefore, for large datasets, searching the entire training set is a very time consuming and computationally expensive task. We believe that we do not have to search the entire set at each iteration.

By using the "59 trick" [75], we propose a selection method, which does not necessitate a full search through the entire dataset but locates an approximate most informative sample by examining a small constant number of randomly chosen samples. The method picks L (L $\ll$ # training instances) random training samples in each iteration and selects the best (closest to the hyperplane) among them. Suppose, instead of picking the closest instance among all the training samples $X_N = (x_1, x_2, \cdots, x_N)$ at each iteration, we first pick a random subset X_L, $L \ll N$ and select the closest sample x_i from X_L based on the condition that x_i is among the top $p\%$ closest instances in X_N with probability $(1-\eta)$. Any numerical modification to these constraints can be met by varying the size of L, and is independent of N. To demonstrate, the probability that at least one of the L instances is among the closest $p\%$ is $1-(1-p\%)^L$. Due to the requirement of $(1-\eta)$ probability, we have

$$1-(1-p\%)^L = 1-\eta \tag{4.1}$$

which follows the solution of L in terms of η and p

$$L = \log \eta \,/\, \log(1-p\%) \tag{4.2}$$

For example, the active learner will pick one instance, with 95% probability, that is among the top 5% closest instances to the hyperplane, by randomly sampling only $\lceil \log(.05)/\log(.95) \rceil = 59$ instances regardless of the training set size. This approach scales well since the size of the subset L is independent of the training

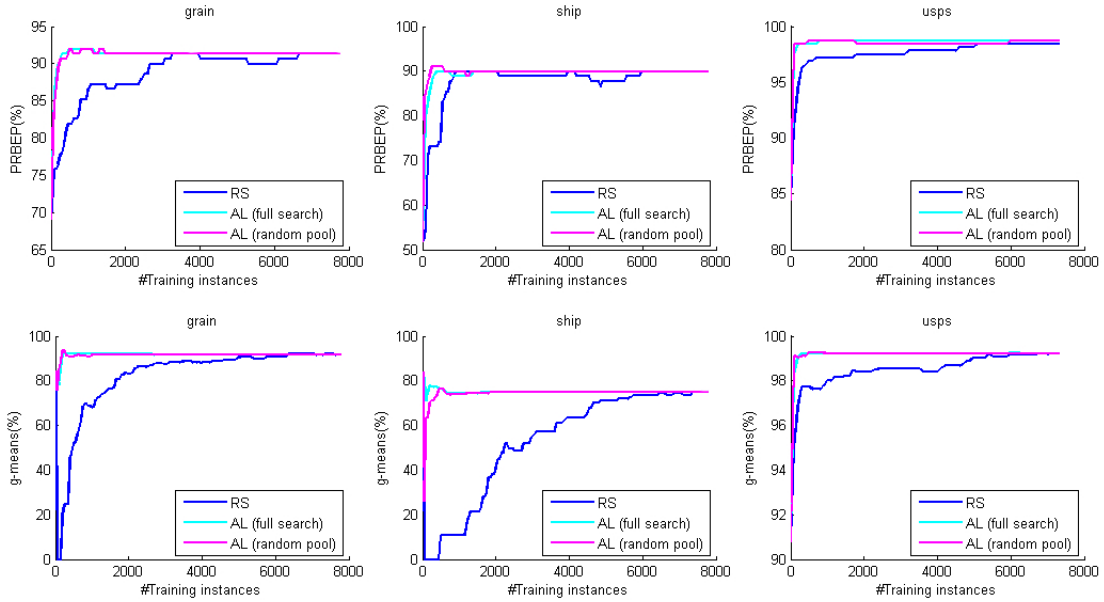


Figure 4.3. Comparison of PRBEP and g-means of RS, AL(full search) and AL(random pool). The comparison of training times of AL(full search) vs. AL(random pool) until saturation in seconds are: 272 vs. 50 (grain), 142 vs. 32 (ship) and 126 vs. 13 (USPS). AL(random pool) is 4 to 10 times faster than AL(full search) with similar prediction performance.

set size N, requires significantly less training time and does not have an adverse effect on the classification performance of the learner.

In our experiments, we set $L = 59$ which means we pick 59 random instances to form the query pool at each learning step and pick the closest instance to the hyperplane from this pool. Figure 4.3 shows the comparisons of PRBEP and g-means performances of the proposed method AL(random pool) and the traditional active learning method AL(full search) [42]. RS corresponds to random sampling where instances are selected randomly. As Figure 4.3 depicts, the proposed active learning method with small pools achieves as good prediction performance as the traditional active learning method. Moreover, the proposed strategy is 4 to 10 times faster than the traditional active learning for the given datasets.

4.2.2 Online SVM for Active Learning

Online learning algorithms are usually associated with problems where the complete training set is not available. However, in cases where the complete training set is available, their computational properties can be leveraged for faster classification and incremental learning. In our framework, we use LASVM instead of a traditional batch SVM tool (e.g., libsvm, SVM^{light}).

Online learning algorithms can select the new data to process either by random or active selection. They can integrate the information of the new seen data to the system without training all the samples again, hence they can incrementally build a learner. This working principle of LASVM leads to speed improvement and less memory demand which makes the algorithm applicable to very large datasets. More importantly, this incremental working principle suits the nature of active learning in a much better way than the batch algorithms. The new informative instance selected by active learning can be integrated to the existing model without retraining all the samples repeatedly. Empirical evidence indicates that a single presentation of each training example to the algorithm is sufficient to achieve training errors comparable to those achieved by the SVM solution [2]. In section 4.2.3 we also show that if we use an early stopping criteria in active sample selection, we do not have to introduce all the training instances to the learner.

4.2.3 Active Learning with Early Stopping

Early stopping criteria is advantageous to the active learning method since it converges to the solution faster than the random sample selection method. A theoretically sound method to stop training is when the examples in the margin are exhausted. To check if there are still unseen training instances in the margin,

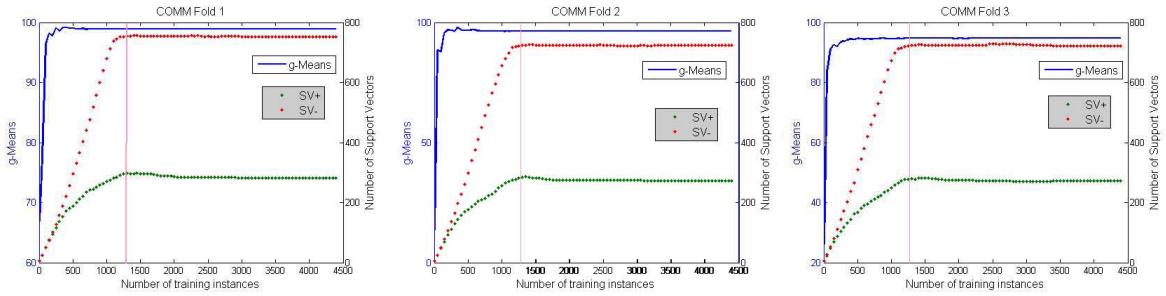


Figure 4.4. 3-fold cross-validation results for the training set of the category COMM in CiteSeer dataset. Vertical lines correspond to early stopping points.

the distance of the new selected instance is compared to the support vectors of the current model. If the new selected instance by active learning (closest to the hyperplane) is not closer than any of the support vectors, we conclude that the margin is exhausted. A practical implementation of this idea is to count the number of support vectors during the active learning training process. If the number of the support vectors stabilizes, it implies that all possible support vectors have been selected by the active learning method.

In order to analyze this method, we conducted a 3-fold cross-validation on one of the datasets (see Figure 4.4). In cross-validation, 2/3 of the training set is used for training and the remaining 1/3 is reserved as the hold-out dataset. Since the training set distribution is representative of the test set distribution, we believe that the algorithm's behavior would most likely be the same in the test set. As can be seen in Figure 4.4, in active learning setups, after using certain number of labeled training data, the number of support vectors saturates and g-means levels off as well. Those graphs support the idea that the model does not change after the system observes enough informative samples. Further, adding more training data after this point does not make a remarkable change in the model and consequently in prediction performance. Notice that in Figure 4.4 the vertical line indicates the suggested early stopping point and it is approximately equal in all three folds. As

Table 4.1. Confusion matrix.

	Actual Positive	Actual Negative
Predicted Positive	TP (true positive)	FP (false positive)
Predicted Negative	FN (false negative)	TN (true negative)

a result, we adopt the early stopping strategy of examining the number of support vectors in the entire training datasets without performing cross-validation.

4.3 Performance Metrics

Classification accuracy is not a good metric to evaluate classifiers in applications with class imbalance problem. SVMs have to achieve a tradeoff between maximizing the margin and minimizing the empirical error. In the non-separable case, if the misclassification penalty C is very small, SVM learner simply tends to classify every example as negative. This extreme approach makes the margin the largest while making no classification errors on the negative instances. The only error is the cumulative error of the positive instances which are already few in numbers. Considering an imbalance ratio of 99 to 1, a classifier that classifies everything as negative, will be 99% accurate but it will not have any practical use as it can not identify the positive instances.

For evaluation of our results, we use several other prediction performance metrics such as g-means, AUC and PRBEP which are commonly used in imbalanced data classification. g-means [69] is denoted as $g = \sqrt{sensitivity \cdot specifity}$ where sensitivity is the accuracy on the positive instances given as $TruePos./(TruePos.+ FalseNeg.)$ and specificity is the accuracy on the negative instances given as $TrueNeg./(TrueNeg. + FalsePos.)$.

The Receiver Operating Curve (ROC) displays the relationship between sensitivity and specificity at all possible thresholds for a binary classification scoring

model, when applied to independent test data. In other words, ROC curve is a plot of the true positive rate against the false positive rate as the decision threshold is changed. The *area under the ROC curve* (AUC) is a numerical measure of a model's discrimination performance and shows how successfully and correctly the model separates the positive and negative observations and ranks them. Since AUC metric evaluates the classifier across the entire range of decision thresholds, it gives a good overview about the performance when the operating condition for the classifier is unknown or the classifier is expected to be used in situations with significantly different class distributions.

Precision Recall Break-Even Point (PRBEP) is another commonly used performance metric for imbalanced data classification. PRBEP is the accuracy of the positive class at the threshold where precision equals to recall. Precision is defined as $TruePos./(TruePos.+FalsePos.)$ and recall is defined as $TruePos./(TruePos.+FalseNeg.)$

4.4 Datasets

We study the performance of the algorithm on various benchmark real-world datasets. The overview of the datasets are given in Table 4.2. The *Reuters-21578* is a collection of newswire stories categorized into hand-labeled topics. We used the "ModApte" split of the *Reuters-21578* dataset that leads to a corpus of 9603 training documents and 3299 test documents. After preprocessing, the training corpus contains 8315 distinct terms. We test the algorithms with 8 of the top 10 most populated categories of *Reuters-21578*. We did not use categories 'earn' and 'acq' since their class imbalance ratios are not high enough. As a text dataset, we also used 5 categories from CiteSeer data. We used 4 benchmark datasets from

Table 4.2. Overview of the datasets.

Dataset		#Feat.	#Pos	#Neg	Ratio	c	γ
Reuters	Crude	8315	389	7381	19.0	2	1
	Grain	8315	433	7337	16.9	2	1
	Interest	8315	347	7423	21.4	1	2
	Money-fx	8315	538	7232	13.4	1	0.5
	Ship	8315	197	7573	38.4	1	0.5
	Wheat	8315	212	7558	35.7	1	0.5
CiteSeer	AI	6946	1420	5353	4.3	50	0.1
	COMM	6946	1252	5341	4.2	50	0.1
	Crypt	6946	552	6041	11.0	50	0.1
	DB	6946	819	5775	7.1	50	0.1
	OS	6946	262	6331	24.2	50	0.1
UCI	Abalone-7	9	352	3407	9.7	100	0.01
	Letter-A	16	710	17290	24.4	10	0.01
	Satimage	36	415	4020	9.69	50	0.001
USPS		256	1232	6097	5.0	1000	2
MNIST-8		780	5851	54149	9.3	1000	0.02

the popular UCI Machine Learning Repository as well. *Letter* and *satimage* are image datasets. The 'letter A' is used as the positive class in *letter* and 'class 4' (damp grey soil) is used as positive class in *satimage*. *Abalone* is a biology dataset. respectively. In *abalone*, instances labeled as 'class 7' are used to form the positive class. *MNIST* and *USPS* are OCR data of handwritten digits and 'digit 8' is used as a positive class in *Mnist*. *Adult* is a census dataset to predict if the income of a person is greater than 50K based on several census parameters, such as age, education, marital status etc. The training set consists of 32,562 instances and the class imbalance ratio is 3. *Waveform* is a popular artificial dataset used commonly in simulation studies. These datasets cover a wide range of data imbalance ratio.

4.5 Experiments and Empirical Evaluation

We first conduct experiments to compare the performance of the proposed active learning strategy AL(random pool) with the traditional active learning method,

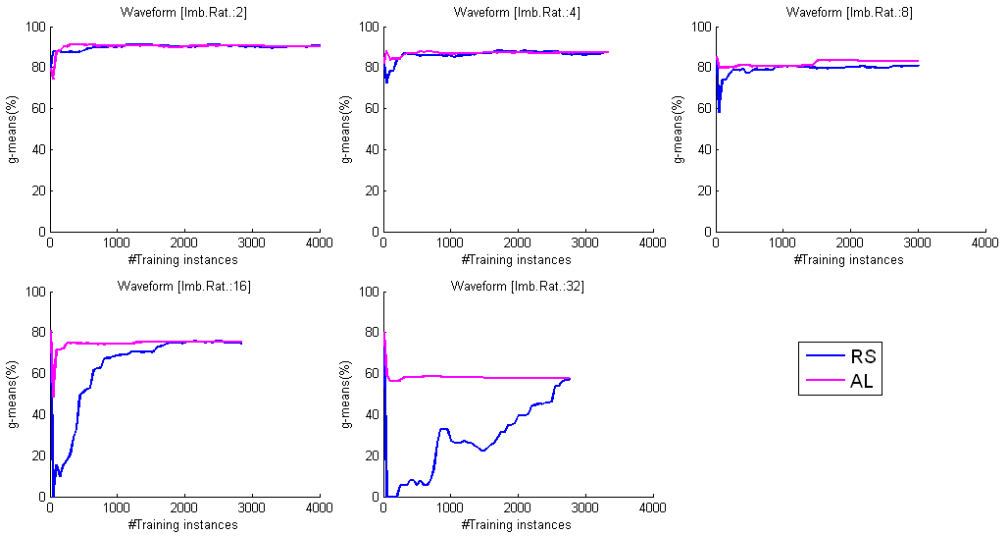


Figure 4.5. Comparison of g-means of AL and RS on the waveform datasets with different imbalance ratios (Imb.R.=2, 4, 8, 16, 32).

AL(full search). The results show that with the proposed method, we can make faster active learning without sacrificing any prediction performance (see Figure 4.3). In the rest of the chapter, we refer to our proposed method as AL since it is the only active learning method that we used afterwards.

In order to make a thorough analysis on the effect of AL to imbalanced data classification, we examine its performance by varying class imbalance ratios using two performance metrics. We randomly remove the instances from the minority class in *Waveform* and *Adult* datasets to achieve different data imbalance ratios. Comparisons of g-means of AL and RS in Figure 4.5 show that the prediction performance of AL is less sensitive to the class imbalance ratio changes than that of the RS. Comparisons of another performance metric PRBEP in Figure 4.6 give even more interesting results. As the class imbalance ratio is increased, AL curves display peaks in the early steps of the learning. This implies that by using an early stopping criteria AL can give higher prediction performance than RS can possibly achieve even after using all the training data. Figure 4.6 curves allow us to think

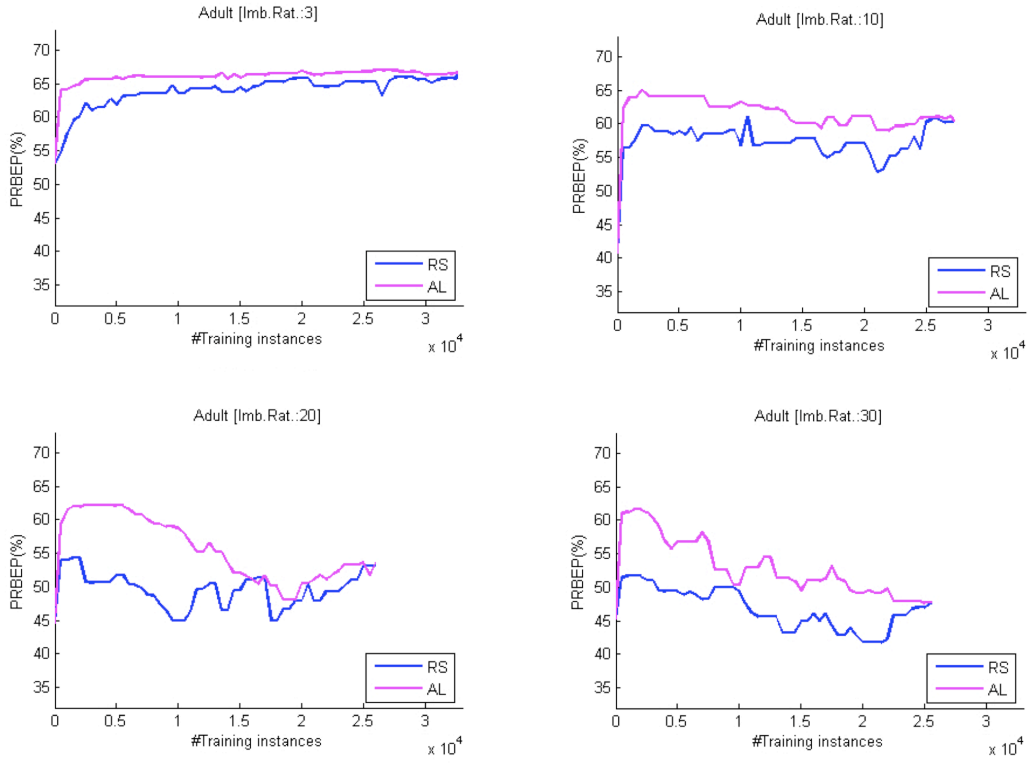


Figure 4.6. Comparison of PRBEP of AL and RS on the adult datasets with imbalance ratios of 3, 10, 20 and 30.

that addition of any instances to the learning model after finding the informative instances can be detrimental to the prediction performance of the classifier. This finding strengthens the idea of applying an early stopping to the active learning algorithms.

We also compared the performance of early stopped AL with Batch algorithm. Table 4.3 presents the g-means and the AUC values of the two methods. Data efficiency column for AL indicates that by processing only a portion of the instances from the training set, AL can achieve similar or even higher prediction performance than that of Batch which sees all the training instances. Another important observation from Table 4.3 is that support vector imbalance ratios in the final models are much less than the class imbalance ratios of the datasets.

Table 4.3. Comparison of g-means and AUC for AL and RS with entire training data (Batch). Support vector ratios are given at the saturation point. Data efficiency corresponds to the percentage of training instances which AL processes to reach saturation.

	Dataset	g-means (%)		AUC (%)		Imb.	SV- /	Data
		Batch	AL	Batch	AL	Rat.	SV+	Efficiency
Reuters	Corn	85.55	86.59	99.95	99.95	41.9	3.13	11.6%
	Crude	88.34	89.51	99.74	99.74	19.0	2.64	22.6%
	Grain	91.56	91.56	99.91	99.91	16.9	3.08	29.6%
	Interest	78.45	78.46	99.01	99.04	21.4	2.19	30.9%
	Money-fx	81.43	82.79	98.69	98.71	13.4	2.19	18.7%
	Ship	75.66	74.92	99.79	99.80	38.4	4.28	20.6%
	Trade	82.52	82.52	99.23	99.26	20.1	2.22	15.4%
	Wheat	89.54	89.55	99.64	99.69	35.7	3.38	11.6%
CiteSeer	AI	87.83	88.58	94.82	94.69	4.3	1.85	33.4%
	COMM	93.02	93.65	98.13	98.18	4.2	2.47	21.3%
	CRYPT	98.75	98.87	99.95	99.95	11.0	2.58	15.2%
	DB	92.39	92.39	98.28	98.46	7.1	2.50	18.2%
	OS	91.95	92.03	98.27	98.20	24.2	3.52	36.1%
UCI	Abalone-7	100.0	100.0	100.0	100.0	9.7	1.38	24.0%
	Letter-A	99.28	99.54	99.99	99.99	24.4	1.46	27.8%
	Satimage	82.41	83.30	95.13	95.75	9.7	2.62	41.7%
	USPS	99.22	99.25	99.98	99.98	4.9	1.50	6.8%
	MNIST-8	98.47	98.37	99.97	99.97	9.3	1.59	11.7%

This confirms our discussion of Figure 4.2 in section 4.2. The class imbalance ratio within the margins are much less than the class imbalance ratio of the entire data and active learning can be used to reach those informative instances which most likely become support vectors without seeing all the training instances.

In order to evaluate the methods at different thresholds, we also investigate the ROC curves as given in Figure 4.7. The ROC curves of AL are similar and sometimes better than of the Batch algorithm (RS, seeing all the training instances). The AUC of AL and Batch are 0.8980 and 0.8910 respectively in the Adult dataset. At the same number of training instances where AL is early stopped, AUC of RS can be substantially lower. As Figure 4.7 shows, the ROC curve of AL is markedly higher than that of RS (early stopping) and the AUC values are 0.8980 and 0.8725

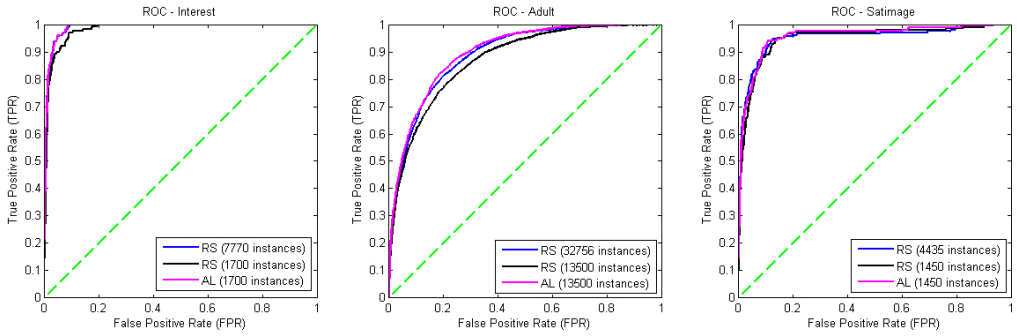


Figure 4.7. Comparison of ROC curves of AL, RS (early stopped at the same number of instances as AL) and RS (with all training data) in Interest, Adult and Satimage datasets.

respectively for Adult dataset. These results suggest that AL converges faster than RS using fewer and informative instances and AL can get even higher prediction performance than the Batch algorithm by processing only a portion of the training set.

In Figure 4.8, we investigate how the number of support vectors changes in AL and Random Sampling (RS). With random sampling, the instances are selected for the learner randomly from the entire pool of the training data. Therefore, the support vector imbalance ratio quickly approaches the data imbalance ratio. As learning continues, the learner should gradually see all the instances within the

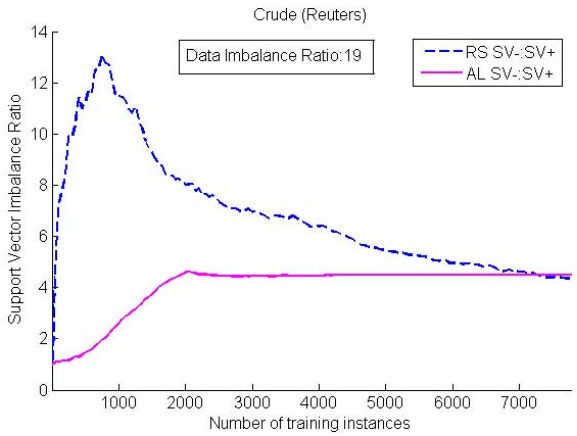


Figure 4.8. Support Vector ratios in AL and RS

final margin and the support vector imbalance ratio decreases. When RS finishes learning, the support vector imbalance ratio is the data imbalance ratio within the margin. The support vector imbalance ratio curve of AL is drastically different than RS. AL intelligently picks the instances closest to the margin in each step. Since the data imbalance ratio within the margin is lower than data imbalance ratio, the support vectors in AL are more balanced than RS during learning. Using AL, the model saturates by seeing only 2000 (among 7770) training instances and reaches the final support vector imbalance ratio. Note that both methods achieve similar support vector imbalance ratios when learning finishes, but AL achieves this in the early steps of the learning.

We compare the AL method discussed here with several other strategies as well. Among them, undersampling (US), and an oversampling method (SMOTE) are examples of resampling techniques which require preprocessing. Recent research showed that oversampling at random does not help to improve prediction performance [76] therefore we use a more complex oversampling method (SMOTE). As an algorithmic method to compare, we use the method of assigning different costs (DC) to the positive and negative classes as the misclassification penalty parameter. For instance, if the imbalance ratio of the data is 19:1 in favor of the negative class, the cost of misclassifying a positive instance is set to be 19 times greater than that of misclassifying a negative one. We use LASVM in all experiments. Other than the results of the methods addressing class imbalance problem, we also give results of Batch algorithm with the original training set to form a baseline. LASVM is run in random sampling mode for US, SMOTE and DC.

We give the comparisons of the methods for g-means performance metric for several datasets in Figure 4.9. The right border of the shaded pink area is the place where the aforementioned early stopping strategy is applied. The curves

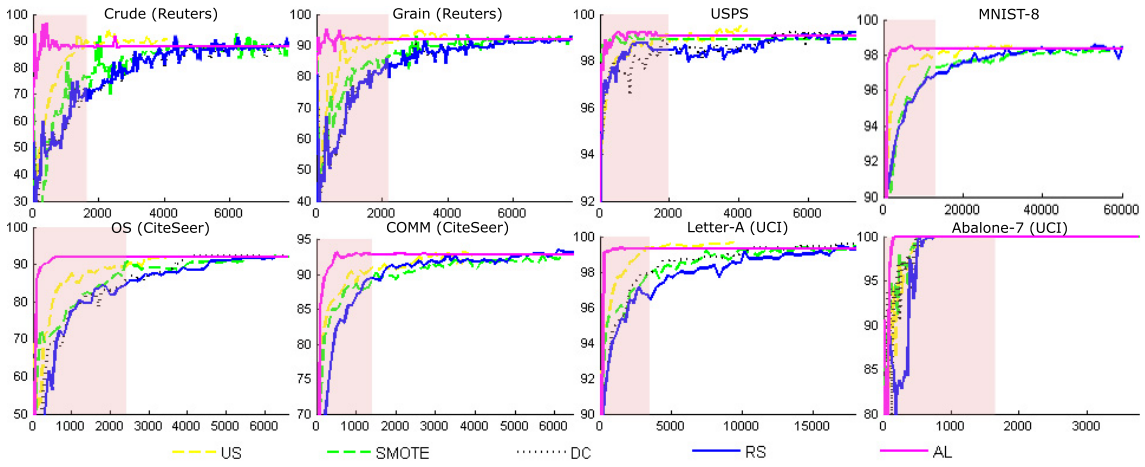


Figure 4.9. Comparisons of g-means. The right border of the shaded area corresponds to the early stopping point.

in the graphs are averages of 10 runs. For completeness we did not stop the AL experiments at the early stopping point but allow them to run on the entire training set. We present the PRBEP of the methods and the total running times of the SMOTE and AL on 18 benchmark and real-world datasets in Table 4.4. The results for active learning in Table 4.4 depict the results in the early stopping points. The results for the other methods in Table 4.4 depict the values at the end of the curves –when trained with the entire dataset– since those methods do not employ any early stopping criteria. We did not apply early stopping criteria to the other methods because as observed from Figure 4.9, no early stopping criteria would achieve a comparable training time with of AL's training time without a significant loss in their prediction performance based on convergence time. The other methods converge to similar levels of g-means when nearly all training instances are used, and applying an early stopping criteria would have little, if any, effect on their training times.

Since AL involves discarding some instances from the training set, it can be perceived as a type of sampling method. Unlike the passive undersampling strategy

Table 4.4. Comparison of PRBEP and training time.

Metric		PRBEP					Training time (sec.)	
Dataset		Batch	US	SMOTE	DC	AL	SMOTE	AL
Reuters	Corn	91.07	78.57	91.07	89.28	89.29	87	16
	Crude	87.83	85.70	87.83	87.83	87.83	129	41
	Grain	92.62	89.93	91.44	91.94	91.94	205	50
	Interest	76.33	74.04	77.86	75.57	75.57	116	42
	Money-fx	73.74	74.30	75.42	75.42	76.54	331	35
	Ship	86.52	86.50	88.76	89.89	89.89	49	32
	Trade	77.77	76.92	77.77	77.78	78.63	215	38
	Wheat	84.51	81.61	84.51	84.51	85.92	54	25
CiteSeer	AI	78.80	80.68	78.99	78.79	79.17	1402	125
	COMM	86.59	86.76	86.59	86.59	86.77	1707	75
	CRYPT	97.89	97.47	97.89	97.89	97.89	310	19
	DB	86.36	86.61	86.98	86.36	86.36	526	41
	OS	84.07	83.19	84.07	84.07	84.07	93	23
UCI	Abalone-7	100.0	100.0	100.0	100.0	100.0	16	4
	Letter-A	99.48	96.45	99.24	99.35	99.35	86	3
	Satimage	73.46	68.72	73.46	73.93	73.93	63	21
USPS		98.44	98.44	98.13	98.44	98.75	4328	13
MNIST-8		97.63	97.02	97.74	97.63	97.74	83,339	1,048

US which discards majority class samples randomly, AL performs an intelligent search for the most informative ones adaptively in each iteration according to the current hyperplane. In datasets where class imbalance ratio is high such as *corn*, *wheat*, *letter* and *satimage*, we observe significant decrease in PRBEP of US (see Table 4.4). Note that US's undersampling rate for the majority class in each category is set to the same value as the final support vector ratio which AL reaches in the early stopping point and RS reaches when it sees the entire training data. Although the class imbalance ratio provided to the learner in AL and US are the same, AL achieves significantly better PRBEP performance metric than US. The Wilcoxon signed-rank test (2-tailed) reveals that the zero median hypothesis can be rejected at the significance level 1% (p=0.0015), implying that AL performs statistically better than US in these 18 datasets. These results reveal

Table 4.5. Comparison of ranks of different methods in PRBEP. The values in bold correspond to the cases where AL win. AL wins in 12 out of 18 cases in PRBEP.

Metric		Rank				
Dataset		Batch	US	SMOTE	DC	AL
Reuters	Corn	1	5	1	4	3
	Crude	1	5	1	1	**1**
	Grain	1	5	4	2	2
	Interest	2	5	1	3	3
	Money-fx	5	4	2	2	**1**
	Ship	4	5	3	1	**1**
	Trade	3	5	3	2	**1**
	Wheat	2	5	2	2	**1**
CiteSeer	AI	4	1	3	5	2
	COMM	3	2	3	3	**1**
	CRYPT	1	5	1	1	**1**
	DB	3	2	1	3	3
	OS	1	5	1	1	**1**
UCI	Abalone-7	1	1	1	1	**1**
	Letter-A	1	5	4	2	2
	Satimage	3	5	3	1	**1**
USPS		2	2	5	2	**1**
MNIST-8		3	5	1	3	**1**
Avg. Rank		2.28	4.00	2.22	2.17	**1.50**

the importance of using the informative instances for learning.

Table 4.5 presents the rank of PRBEP prediction performance of the five approaches in a variety of datasets. The values in bold correspond to the cases where AL wins and it's clear that winning cases are very frequent for AL (12 out of 18 cases). The average rank also indicates that AL achieves the best PRBEP among the five methods. SMOTE and DC achieve higher PRBEP than the Batch algorithm. The loss of information when undersampling the majority class affects US's prediction performance. Table 4.4 also gives the comparison of the computation times of the AL and SMOTE. Note that SMOTE requires significantly long preprocessing time which dominates the training time in large datasets, e.g., MNIST-8 dataset. The low computation cost, scalability and high prediction performance of

AL suggest that AL can efficiently handle the class imbalance problem.

4.6 Remarks

The class imbalance problem has been known to be detrimental to the prediction performance of classification algorithms. The results offer a better understanding of the effect of the active learning on imbalanced datasets. We first propose an efficient active learning method which selects informative instances from a randomly picked small pool of examples rather than making a full search in the entire training set. This strategy renders active learning to be applicable to very large datasets which otherwise would be computationally very expensive. Combined with the early stopping heuristics, active learning achieves a fast and scalable solution without sacrificing prediction performance. We then show that the proposed active learning strategy can be used to address the class imbalance problem. In simulation studies, we demonstrate that as the imbalance ratio increases, active learning can achieve better prediction performance than random sampling by only using the informative portion of the training set. By focusing the learning on the instances around the classification boundary, more balanced class distributions can be provided to the learner in the earlier steps of the learning. Our empirical results on a variety of real-world datasets allow us to conclude that active learning is comparable or even better than other popular resampling methods in dealing with imbalanced data classification.

Chapter 5

Adaptive Oversampling for Imbalanced Data Classification

"Not what we have, but what we enjoy, constitutes our abundance."

John Petit-Senn

In the previous chapter, we outlined the characteristics of imbalanced datasets and discussed the ways in which active learning can be leveraged to address the classification problems with imbalanced data. This chapter presents a resampling method, namely VIRTUAL (**V**irtual **I**nstances **R**esampling **T**echnique **U**sing **A**ctive **L**earning) [4], for classification tasks in imbalanced datasets. VIRTUAL leverages the power of active learning to intelligently and adaptively oversample the data. VIRTUAL generates virtual instances for the minority class support vectors during the training process, therefore it removes the need for an extra preprocessing stage. The method also uses an efficient active learning strategy to pick the informative samples from the training set in the early stages of the learning process. Informative samples are more likely to become support vectors in the SVM model.

Since support vectors are found in the early stages of the learning, corresponding informative virtual examples will also be created in the early stages. This will eliminate the algorithm of creating excessive and redundant virtual instances. Empirical evaluation shows that as an adaptive method, VIRTUAL achieves superior prediction performance than its components. In addition, VIRTUAL is more efficient in generating new instances due to its adaptive nature in the learning steps. We also present that it has a shorter training time than other competitive oversampling techniques.

5.1 VIRTUAL Algorithm

VIRTUAL is an adaptive oversampling method for creating virtual instances for the minority class in imbalanced datasets. Traditional oversampling techniques act as an *offline* step that generate virtual instances of the minority class prior to the training process. After generating the virtual instances in the preprocessing stage, SVM then runs in batch mode and learns a model using all instances in this augmented training set. However, since not all instances in the training set are equally informative, we believe that there is no need to create virtual instances for each positive instance, as they may increase the redundancy and burden the learner. Instead, we can intelligently create virtual instances according to the informativeness of a sample determined by their distance to the hyperplane. The proposed VIRTUAL method creates virtual positive instances in the proximity of the support vectors. The method uses the online LASVM algorithm where its model is continually modified as it sequentially processes training instances. This working principle of LASVM leads to speed improvement and less memory requirements, which makes the algorithm applicable to very large datasets. More importantly,

this incremental working principle suits the nature of active learning in a much better way than batch algorithms. The new informative instance selected by active learning can be integrated to the existing model without retraining all the samples repeatedly.

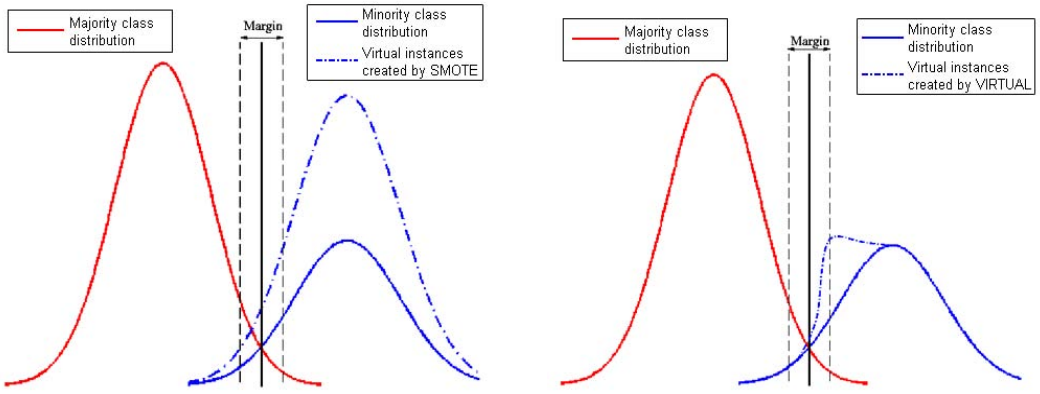
We first briefly discuss the active learning step of VIRTUAL and explain the virtual instance generation process of the algorithm. We then highlight the advantages of the proposed method.

5.1.1 Active Selection of Instances

Let S denote the pool of real and virtual training examples unseen by the learner at each active learning step. Active learning is an iterative process that, at each iteration, queries the instances in S and selects the most informative instance to the learner. Instead of searching for the most informative instance among all the samples in S, VIRTUAL queries a randomly picked smaller pool from S, based on the querying scheme described in section 4.2.1. From the small pool, VIRTUAL selects an instance that is closest to the hyperplane according to the current model. If the selected instance is a real positive instance (from the original training data) and it becomes a support vector, VIRTUAL advances to the oversampling step, which we explain in the following section. Otherwise, the algorithm proceeds to the next iteration to select another instance.

5.1.2 Virtual Instance Generation

We propose an oversampling technique that creates virtual instances within the k nearest minority class neighbors of the minority class support vectors. VIRTUAL oversamples the real minority class instance (from the original training data) which



(a) Oversampling with SMOTE (b) Oversampling with VIRTUAL

Figure 5.1. Comparison of oversampling the minority class using SMOTE and VIRTUAL.

become a support vector in the current iteration. It selects the k nearest minority class neighbors $(x_{i\to 1} \cdots x_{i\to k})$ of x_i based on their similarities in the kernel transformed higher dimensional feature space. We notice that little variations in k does not cause dramatic changes in the performance and we set k to 5 to be consistent with SMOTE. We limit the neighboring instances of x_i to the minority class so that the new virtual instances lie within the minority class distribution. Depending on the amount of over-sampling required, the algorithm creates v virtual instances. Since the class distribution within the margin is not that imbalanced (see Figure 5.1), a large value of v is unnecessary, therefore we set v as 1 in both VIRTUAL and SMOTE to have an oversampling ratio of 2. Each virtual instance lies on any of the line segments joining x_i and its neighbor $x_{i\to j}$ $(j = 1, ..., k)$. In other words a neighbor $x_{i\to j}$ is randomly picked and the virtual instance is created as $\bar{x}_v = \lambda \cdot x_i + (1 - \lambda)x_{i\to j}$, where $\lambda \in (0, 1)$ determines the placement of $\bar{x}_v$ between x_i and $x_{i\to j}$. All v virtual instances are added to S and are eligible to be picked by the active learner in the subsequent iterations.

Algorithm 5.1 VIRTUAL

Define:
$X = \{x_1, x_2, \cdots, x_n\}$: training instances
X_R^+ : positive real training instances
S : pool of training instances for SVM
v : # virtual instances to create in each iteration
L : size of the small set of randomly picked samples for active sample selection

```
1.  Initialize S ← X
2.  while S ≠ ∅
3.      //Active sample selection step
4.      d_min ← ∞
5.      for i ← 1 to L
6.          x_j ← RandomSelect(S)
7.          If d(x_j, hyperplane) < d_min
8.              d_min ← d(x_j, hyperplane)
9.              candidate ← x_j
10.         end
11.     end
12.     x_s ← candidate
13.     //Virtual Instance Generation
14.     If x_s becomes SV and x_s ∈ X_R^+
15.         K ← k nearest neighbors of x_s
16.         for i ← 1 to v
17.             x_m ← RandomSelect(K)
18.             //Create a virtual positive instance x_{s,m}^v between x_s and x_m
19.             λ=random number between 0 and 1
20.             x_{s,m}^v = λ · x_s + (1 − λ)x_m
21.             S ← S ∪ x_{s,m}^v
22.         end
23.     end
24.     S ← S − x_s
25. end
```

The pseudocode of VIRTUAL is given in Algorithm 5.1 and depicts the two processes described above. In the beginning, the pool S contains all real instances in the training set. At the end of each iteration, the instance selected is removed from S, and any virtual instances generated are included in the pool S. VIRTUAL terminates when there are no instances in S. In Section 5.2.1, we propose an early stopping criteria for VIRTUAL.

5.1.3 Remarks on VIRTUAL

We compare VIRTUAL with a popular oversampling technique SMOTE to illustrate the advantages of the proposed algorithm. Figure 5.1 shows the different behavior of how SMOTE and VIRTUAL create virtual instances for the minority class. SMOTE creates virtual instance(s) for each positive example (see Figure 5.1(a)), whereas VIRTUAL creates the majority of virtual instances around the positive canonical hyperplane (shown with a dashed line in Figure 5.1(b)). Note that a large portion of virtual instances created by SMOTE are far away from the hyperplane and thus are not likely to be selected as support vectors. VIRTUAL, on the other hand, generates virtual instances near the real positive support vectors adaptively in the learning process. Hence the virtual instances are near the hyperplane and thus are more informative.

We further analyze the computation complexity of SMOTE and VIRTUAL. The computation complexity of VIRTUAL is $O(|SV(+)| \cdot v \cdot \mathcal{C})$, where v is the number of virtual instances created for a real positive support vector in each iteration, $|SV(+)|$ is the number of positive support vectors and $\mathcal{C}$ is the cost of finding k nearest neighbors. The computation complexity of SMOTE is $O(\left|X_R^+\right| \cdot v \cdot \mathcal{C})$, where $\left|X_R^+\right|$ is the number of positive training instances. $\mathcal{C}$ depends on the approach for finding k nearest neighbors. The naive implementation searches all N training instances for the nearest neighbors and thus $\mathcal{C} = kN$. Using advanced data structure such as kd-tree, $\mathcal{C} = k \log N$. Since $|SV(+)|$ is typically much less than $\left|X_R^+\right|$, VIRTUAL incurs lower computation overhead than SMOTE. Also, with fewer virtual instances created, the learner is less burdened with VIRTUAL. We demonstrate with empirical results that the virtual instances created with VIRTUAL are more informative and the prediction performance is also improved.

5.2 Experiments

We conducted a series of experiments on both artificial and real-world datasets to demonstrate the efficacy of the proposed algorithm. VIRTUAL is compared to two systems, Active Learning (AL) and Synthetic Minority Oversampling Technique (SMOTE). AL solely adopts the aforementioned active learning strategy without preprocessing or creating any virtual instances during learning. On the other hand, SMOTE preprocesses the data by creating virtual instances before training and uses random sampling in learning. Experiments elicit the advantages of adaptive virtual sample creation in VIRTUAL.

5.2.1 Simulation Study

The popular waveform example is widely used in classification literature [77]. In this example, three shifted triangular waveforms

$$\begin{aligned} v_1(j) &= max(6 - |j - 11|, 0) \\ v_2(j) &= v_1(j - 4) \\ v_3(j) &= v_1(j + 4) \end{aligned}$$

are linearly combined into 21 variables:

$$x_j = \begin{cases} u \cdot v_1(j) + (1 - u)v_2(j) + \varepsilon_j, \text{ Class 1} \\ u \cdot v_1(j) + (1 - u)v_3(j) + \varepsilon_j, \text{ Class 2} \\ u \cdot v_2(j) + (1 - u)v_3(j) + \varepsilon_j, \text{ Class 3} \end{cases}$$

where $j = 1, 2, ..., 21$, u is uniformly distributed on $(0, 1)$ and ε_j is the normal Gaussian noise. In our case, we use samples from Class 1 as positive samples and

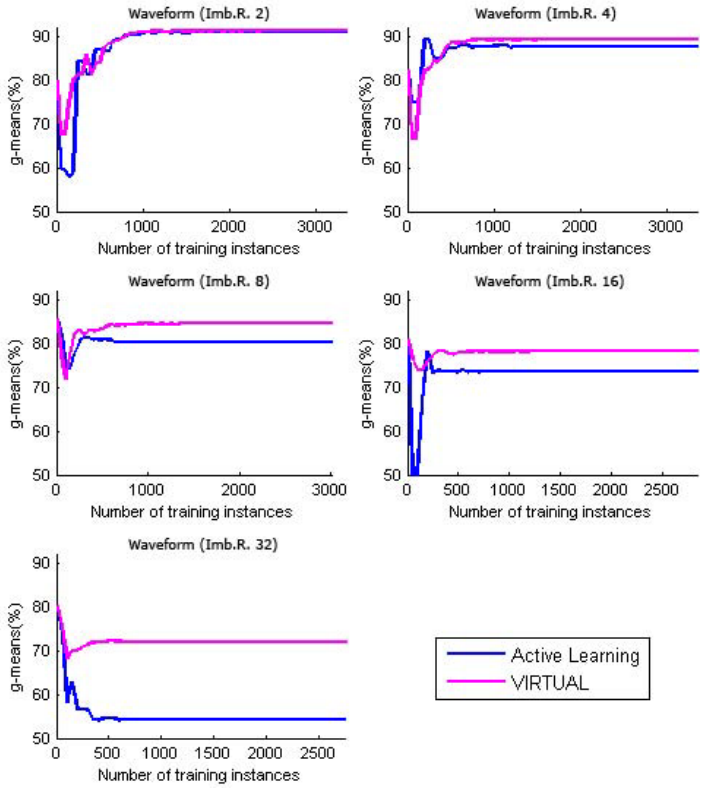


Figure 5.2. Comparison of Active Learning and VIRTUAL on the *Waveform* datasets with different imbalance ratios (Imb.R.=2, 4, 8, 16, 32). The test results are average of ten runs.

the others as negative and thus the imbalance ratio is 2. We randomly draw 4,000 samples for training and independently draw 1,000 samples for testing.

To showcase the different behavior of AL and VIRTUAL in different imbalance ratio (Imb.R.) data, we undersample the positive class to obtain five datasets (*Waveform* Imb.R.= $2^1, 2^2, 2^3, 2^4, 2^5$). Figure 5.2 demonstrates the g-means of AL and VIRTUAL in these five datasets. We observe that when data is moderately imbalanced (Imb.R.=2, 4), the g-means curves of the two methods are close to each other. When the imbalance ratio increases, the g-means of AL drops faster than that of VIRTUAL and thus the gap widens. In the *Waveform* Imb.R. 32 dataset, there are 83 positive instances and AL uses them all as support vectors. Since

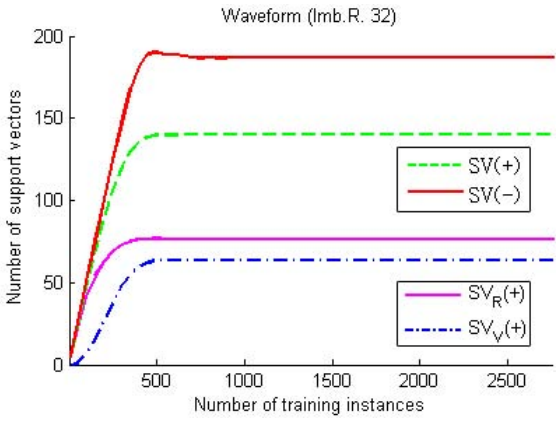


Figure 5.3. Number of support vectors in VIRTUAL versus number of training instances in *Waveform* (Imb.R.=32). Note that $SV(+) = SV_R(+) + SV_V(+)$.

VIRTUAL (v=1) creates a virtual instance for each real positive support vector, it creates 83 positive virtual instances and most of them are selected as support vectors (see Figure 5.3). As a result, the number of positive support vectors nearly doubles and the number of positive and negative support vectors is more balanced. Figure 5.2 compares the g-means of AL and VIRTUAL for different class imbalance ratios and illustrates that these adaptively created positive training instances in VIRTUAL contribute to the improvement in g-means over AL. Therefore, VIRTUAL is shown to be more resilient to imbalanced data.

Figure 5.3 further unpacks the adaptive learning process of VIRTUAL in the *Waveform* Imb.R. 32 dataset. When the first 150 training instances are seen, the number of real positive support vectors ($SV_R(+)$) and negative support vectors ($SV(-)$) is balanced. This effect is due to the active learning strategy, because in early iterations active learning picks positive and negative samples in a balanced manner whereas random sampling selects examples proportional to the data imbalance ratio. In this stage, the number of virtual positive instances is relatively small compared to the real positive instances and thus few are selected into the model. Later, the real positive samples are selected more slowly (deviating from

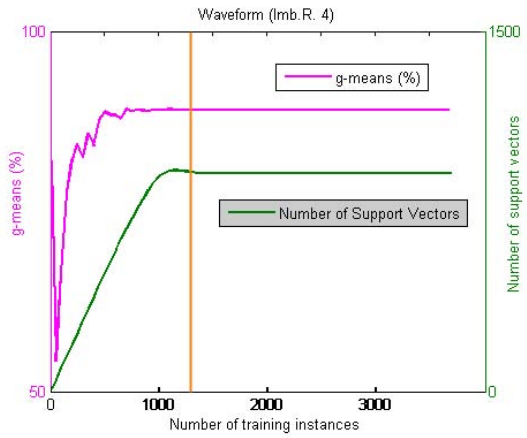


Figure 5.4. Saturation of number of support vectors and g-means for *Waveform* (Imb.R.=4). The vertical line indicates where support vectors saturate and training stops.

the straight line) and they are eventually exhausted. During this stage, virtual positive instances continue to be created and a large portion of them are selected into the model. Finally, the model becomes saturated using only 500 of the original total 2,757 training instances. By creating virtual examples, VIRTUAL nearly doubles the number of positive support vectors and thus the support vector ratio becomes more balanced. As seen in Figure 5.2, the adaptively created virtual examples help to guide the learner to achieve better g-means. This experiments on artificial dataset demonstrates the superiority of VIRTUAL over AL in more imbalanced datasets.

In Figure 5.4, we observe that when the number of support vectors saturates g-means also stabilizes. Seeing additional instances after a point does not change the model, as the training instances in the margin are exhausted. Accordingly, we apply an early stopping criteria to eliminate the learning stage which has little, if any, impact on the prediction performance. A theoretically sound method to stop training is to check if there are still unseen training instances in the margin, the distance of the newly selected instance is compared to the support vectors

Table 5.1. The Reuters and 4 UCI datasets.

Dataset		#Feature	#Pos	#Neg	Imb. Ratio
Reuters	acq	8315	1650	6120	3.7
	corn	8315	181	7589	41.9
	crude	8315	389	7381	19.0
	earn	8315	2877	4893	1.7
	grain	8315	433	7337	16.9
	interest	8315	347	7423	21.4
	money-fx	8315	538	7232	13.4
	ship	8315	197	7573	38.4
	trade	8315	369	7401	20.1
	wheat	8315	212	7558	35.7
UCI	abalone	9	352	3407	9.7
	breast	9	172	320	1.9
	letter	16	710	17290	24.4
	satimage	36	415	4020	9.7

of the current model. If the new selected instance by active learning (closest to the hyperplane) is not closer than any of the support vectors, we conclude that the margin is exhausted. A practical implementation of this idea is to count the number of support vectors during the active learning process. If the number of the support vectors stabilizes, it implies that all possible support vectors have been selected into the model. Early stopping shortens the training time without sacrificing prediction performance. We adopt this strategy in our experiments to find the early stopping points where active learning is used.

5.2.2 Experiments on Real-World Data

We study the performance of the algorithm on real-world data using several benchmark datasets. The *Reuters-21578* is a popular text mining benchmark dataset. We test the algorithms with the top 10 most populated categories of *Reuters-21578*. In each category relevant instances are labeled as positive and the remaining as negative. We also used 4 benchmark datasets from the popular UCI Machine

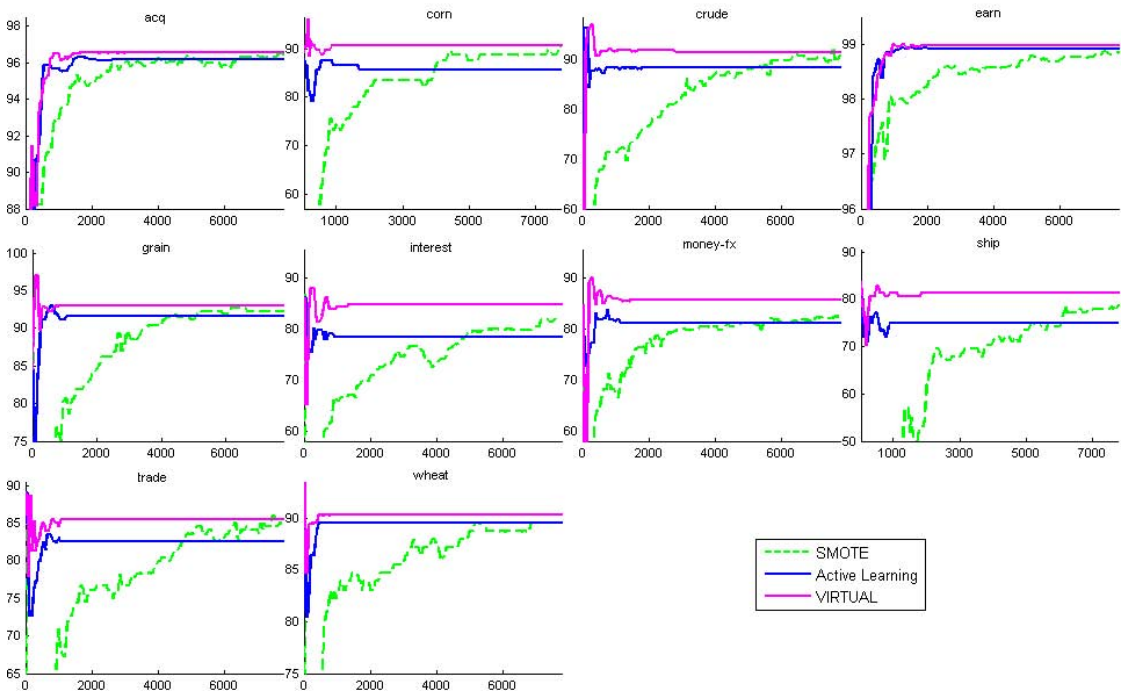


Figure 5.5. Comparison of SMOTE, AL and VIRTUAL on 10 largest categories of *Reuters-21578*. We show the g-means (%) (y-axis) of the current model for the test set versus the number of training samples (x-axis) seen.

Learning Repository. *Letter* and *satimage* are image datasets. The 'letter A' is used as the positive class in *letter* and 'class 4' (damp grey soil) is used as positive class in *satimage*. *Abalone* and *Wisconsin breast cancer* (*breast*) are biology and medical diagnosis datasets respectively. In *abalone*, instances labeled as 'class 7' form the positive class and instances labeled as 'malignant' constitute the positive class in *breast*. These datasets cover a wide range of data imbalance ratio (see Table 5.1).

In Figures 5.5 and 5.6, we provide the details on the behavior of the three algorithms, SMOTE, AL and VIRTUAL. For the Reuters datasets (Figure 5.5), we note that in all the 10 categories VIRTUAL outperforms AL in g-means metric after saturation. The difference in performance is most pronounced in the more imbalanced categories, e.g. *corn*, *interest* and *ship*. In the less imbalanced datasets such

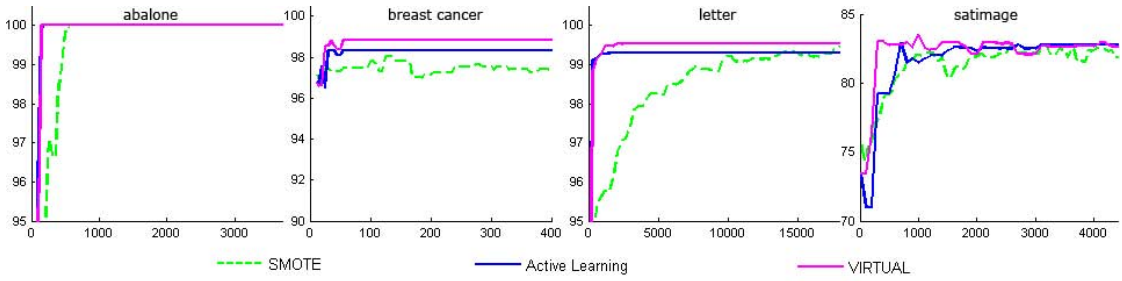


Figure 5.6. Comparison of SMOTE, AL and VIRTUAL on *UCI* datasets. We present the g-means (%) (y-axis) of the current model for the test set vs. the number of training samples (x-axis) seen.

as *acq* and *earn*, the difference in g-means of both methods is less noticeable. The g-means of SMOTE converges much slower than both AL and VIRTUAL. However, SMOTE converges to higher g-means than AL in some of the categories, indicating that the virtual positive examples provide additional information that can be used to improve the model. VIRTUAL converges to the same or even higher g-means than SMOTE while generating fewer virtual instances. For the UCI datasets (Figure 5.6), VIRTUAL performs as well as AL in *abalone* in g-means and consistently outperforms AL and SMOTE in the other three datasets.

In Table 5.2, the support vector imbalance ratio of all the three methods are lower than the data imbalance ratio, and VIRTUAL achieves the most balanced ratios of positive and negative support vectors in the Reuters datasets. Despite that the datasets we used have different data distributions, the portion of virtual instances which become support vectors in VIRTUAL consistently and significantly higher than that in SMOTE. These results confirm our previous discussion that VIRTUAL is more effective in generating informative virtual instances.

Table 5.3 presents g-means and the total learning time for SMOTE, AL and VIRTUAL. Classical batch SVM's g-means values are also provided as a reference point. In Reuters datasets, VIRTUAL yields the highest g-means in all categories.

Table 5.2. Support vectors with SMOTE (SMT), AL and VIRTUAL. Imb.Rt. is the data imbalance ratio and #SV(-)/#SV(+) represents the support vector imbalance ratio. The rightmost two columns compare the portion of the virtual instances selected as support vectors in SMOTE and VIRTUAL.

Dataset		Imb. Rt.	#SV(-)/#SV(+)			$\#SV_V(+)/\#$V.I.	
			SMT	AL	VIRTUAL	SMT	VIRTUAL
Reuters	acq	3.7	1.24	1.28	1.18	2.4%	**20.3%**
	corn	41.9	2.29	3.08	1.95	17.1%	**36.6%**
	crude	19.0	2.30	2.68	2.00	10.8%	**50.4%**
	earn	1.7	1.68	1.89	1.67	6.0%	**24.2%**
	grain	16.9	2.62	3.06	2.32	7.2%	**42.3%**
	interest	21.4	1.84	2.16	1.66	13.3%	**72.2%**
	money-fx	13.4	1.86	2.17	1.34	8.2%	**31.1%**
	ship	38.4	3.45	4.48	2.80	20.0%	**66.5%**
	trade	20.1	1.89	2.26	1.72	15.4%	**26.6%**
	wheat	35.7	2.55	3.43	2.22	12.3%	**63.9%**
UCI	abalone	9.7	0.99	1.24	0.99	30.4%	**69.2%**
	breast	1.9	1.23	0.60	0.64	2.9%	**39.5%**
	letter	24.4	1.21	1.48	0.97	0.98%	**74.4%**
	satimage	9.7	1.31	1.93	0.92	37.3%	**53.8%**

Table 5.3. g-means and total learning time using SMOTE, AL and VIRTUAL. 'Batch' corresponds to the classical SVM learning in batch setting without resampling. The numbers in brackets denote the rank of the corresponding method in the dataset.

Dataset		g-means (%)				Total learning time (sec.)		
		Batch	SMOTE	AL	VIRTUAL	SMOTE	AL	VIRTUAL
Reuters	acq	96.19 (3)	96.21 (2)	96.19 (3)	**96.54 (1)**	2271	146	203
	corn	85.55 (4)	89.62 (2)	86.59 (3)	**90.60 (1)**	74	43	66
	crude	88.34 (4)	91.21 (2)	88.35 (3)	**91.74 (1)**	238	113	129
	earn	98.92 (3)	**98.97 (1)**	98.92 (3)	**98.97 (1)**	4082	121	163
	grain	91.56 (4)	92.29 (2)	91.56 (4)	**93.00 (1)**	296	134	143
	interest	78.45 (4)	83.96 (2)	78.45 (4)	**84.75 (1)**	192	153	178
	money-fx	81.43 (3)	83.70 (2)	81.08 (4)	**85.61 (1)**	363	93	116
	ship	75.66 (3)	78.55 (2)	74.92 (4)	**81.34 (1)**	88	75	76
	trade	82.52 (3)	84.52 (2)	82.52 (3)	**85.48 (1)**	292	72	131
	wheat	89.54 (3)	89.50 (4)	89.55 (2)	**90.27 (1)**	64	29	48
UCI	abalone	**100 (1)**	**100 (1)**	**100 (1)**	**100 (1)**	18	4	6
	breast	98.33 (2)	97.52 (4)	98.33 (2)	**98.84 (1)**	4	1	1
	letter	99.28 (3)	99.42 (2)	99.28 (3)	**99.54 (1)**	83	5	6
	satimage	**83.57 (1)**	82.61 (4)	82.76 (3)	82.92 (2)	219	18	17

Table 5.3 shows the effectiveness of adaptive virtual instance generation. In categories *corn*, *interest* and *ship* with high class imbalance ratio, VIRTUAL gains substantial improvement in g-means. Compared to AL, VIRTUAL requires additional time for the creation of virtual instances and selection of those which may become support vectors. Despite this overhead, VIRTUAL's training times are comparable with that of AL. In the cases where minority examples are abundant, SMOTE demands substantially longer time to create virtual instances than VIRTUAL. But as the rightmost columns in Table 5.2 show, only a small fraction of the virtual instances created by SMOTE become support vectors. Therefore SMOTE spends much time to create virtual instances that will not be used in the model. On the other hand, VIRTUAL has already a short training time and uses this time to create more informative virtual instances. In Table 5.3, the numbers in parentheses give the ranks of the g-means prediction performance of the four approaches. The values in bold correspond to a win and VIRTUAL wins in nearly all datasets. The Wilcoxon signed-rank test (2-tailed) between VIRTUAL and its nearest competitor SMOTE reveals that the zero median hypothesis can be rejected at the significance level 1% ($p = 4.82 \times 10^{-4}$), implying that VIRTUAL performs statistically better than SMOTE in these 14 datasets. These results reveal the importance of creating synthetic samples from the informative instances rather than all the instances.

5.3 Remarks

We propose a novel oversampling technique VIRTUAL to address the imbalanced data classification problem in SVMs. Rather than creating virtual instances for each positive instance as in conventional oversampling techniques, VIRTUAL adaptively creates instances according to the real positive support vectors selected in

each active learning step. These instances are informative as they are close to the hyperplane. Thus, VIRTUAL creates fewer virtual instances that are informative. Our complexity analysis shows that VIRTUAL incurs lower overhead in data generation and eventually less burden to the learner. Our thorough empirical results on both artificial and real-world data demonstrate that VIRTUAL is capable of achieving higher g-means than active learning without oversampling (AL) and SMOTE. Experimental results also show that VIRTUAL is more resilient to high class imbalance ratios due to its capability of creating more balanced models using the virtual instances created. The training time of VIRTUAL is substantially shorter than SMOTE in most cases.

The proposed framework focuses on class imbalance classification problem in SVMs. A similar approach is applicable to other machine learning paradigms such as boosting, which implicitly maximizes the margin.

Chapter 6

Non-Convex Online Support Vector Machines

"The greatest enemy of knowledge is not ignorance, it is the illusion of knowledge."

Stephen Hawking

In supervised learning systems, the generalization performance of classification algorithms are shown to be greatly improved with large margin training. Large margin classifiers find the maximal margin hyperplane that separates the training data in the appropriately chosen kernel induced feature space. It has been shown numerous times that if a large margin is obtained, the separating hyperplane is likely to have a small misclassification rate during recognition (or prediction) [78, 17, 79]. The maximal margin methodology forms the fundamental principles of Support Vector Machines (SVMs). In the presence of noise, however, the standard maximum margin algorithm can be subject to overfitting. Cortes and Vapnik [10] address this problem by proposing the soft margin criterion, which allows some misclassified examples in the training phase for better predictive power. However,

the soft margin approach in SVMs has brought a serious shortcoming along with its advantages. With the soft margin criterion, patterns are allowed to be misclassified for a certain cost and outlier (misclassified) examples start to play a dominant role in determining the decision hyperplane, since they tend to have the largest margin loss according to the Hinge Loss. Nonetheless, due to its convex property and practicality, Hinge Loss became a commonly used loss function in SVMs.

Convexity is viewed as a virtue by most of the machine learning researchers both from a theoretical and experimental point of view. Convex methods can easily be mathematically analyzed and bounds can be produced. Additionally, convex solutions are guaranteed to reach to the global optimum, avoiding the fear of ending up in the local optimum. The popularity of convexity further increased after the success of convex algorithms, particularly with SVMs, which yield good generalization performance and strong theoretical foundations. However, despite many advantages of convex modeling, the price we pay for insisting on convexity is an increase in the size of the model and the scaling properties of the algorithm. In this chapter, we show that shifting gears from convexity to non-convexity can be very effective for achieving sparse and scalable solutions, particularly when the data consists of abundant label noise.

The quality of a dataset can be characterized by its instances' *attributes* and *class labels*. The former indicates how well the attributes characterize instances for classification purpose, and the latter represents whether the class of each instance is correctly assigned [80]. The class labels of the instances can be incorrectly labeled due to subjectivity, data entry error, or inadequacy of the information used to label each instance [81]. Both attributes and class labels can be a source of *noise*, which can reduce system performance in terms of classification accuracy, time in building a classifier and the size of the classifier. This work particularly

addresses noise within the context of incorrect class labels. We present herein experimental results showing how a non-convex loss function, *Ramp Loss*, can be efficiently integrated to an online SVM algorithm in order to suppress the influence of instances with class label noise.

Various works in the history of machine learning research focused on using non-convex loss functions as an alternate to convex Hinge Loss, in large margin classifiers. While Mason et al. [82] and Krause and Singer [83] applied it to Boosting, Perez-Cruz et al. [84] and Linli Xu [85] proposed training algorithms for SVMs with the Ramp Loss and solved the non-convex optimization by utilizing semi-definite programming and convex relaxation techniques. On the other hand, some previous work of Liu et al. [86] and Wang et al. [87] used the concave-convex programming (CCCP) for non-convex optimization as the work presented here. Those studies are worthwhile in the endeavor of achieving sparse models or competitive generalization performance, nevertheless none of them are efficient in terms of computational running time and scalability for real-world data mining applications and yet the improvement in classification accuracy is only marginal. Collobert et al. [88] pointed out the scalability advantages of non-convex approaches and used CCCP for non-convex optimization in order to achieve faster batch SVMs and Transductive SVMs. This chapter focuses on bringing the scalability advantages of non-convexity to the online learning setting by using the LASVM algorithm that was presented in Chapter 3.

Online learning is advantageous when dealing with streaming or very large scale data. Online learners incorporate the information of new seen training data into the model without retraining it with the previously seen entire training data. Since they process the data one at a time in the training phase, selective sampling can be applied and evaluation of the informativeness of the data prior to the

processing by the learner becomes possible. This chapter presents an online SVM algorithm with non-convex loss function (LASVM-NC) [5], which yields a significant speed improvement in training and builds a sparser model, hence resulting in faster recognition than its convex version as well. Based on selective sampling, we further propose an SVM algorithm (LASVM-I) [5] that ignores the instances that lie in the flat region of the Ramp Loss in advance, before they are processed by the learner. Although this may appear like an over-aggressive training sample elimination process, we point out that those instances do not play role in determining the decision hyperplane according to the Ramp Loss anyway. Making a right decision about whether to eliminate or process a training data highly depends on the trustworthiness of the current model. The intermediate models should be well enough trained in order to capture the characteristics of the training data, but on the other hand, should not be over-optimized since only part of the entire training data is seen at that point in time. We build a balance within those two situations by leveraging the gap between primal and dual functions during the optimization steps of online SVM (LASVM-G) [5]. We then build a non-convex optimization scheme and a training sample ignoring mechanism on top of LASVM-G. We show that for a particular case of sample elimination scenario, misclassified instances according to the current learned model are not taken into account at the training process ($s = 0$). For another case, only the instances in the margin pass the barrier of elimination and are processed in the training, hence leading to an extreme case of *small pool active learning* framework [3] in online SVMs (when $s = -1$).

The proposed non-convex implementation and selective sample ignoring policy yields sparser models with fewer support vectors and faster training with less computational time and kernel computations which overall leads to a more scalable online SVM algorithm. The advantages of the proposed methods become more

pronounced in noisy data classification where mislabeled samples are in abundance.

6.1 Gap-based Optimization – LASVM-G

LASVM-G is an efficient online SVM algorithm that brings performance enhancements to LASVM. Instead of running a single REPROCESS operation after each PROCESS step, LASVM-G adjusts the number of REPROCESS operations at each online iteration by leveraging the gap between the primal and the dual functions. Further, LASVM-G replaces LASVM's one time FINISHING optimization and cleaning stage with the optimizations performed in each REPROCESS cycle at each iteration and the periodic non-SV removal steps. These improvements enable LASVM-G to generate more reliable intermediate models than LASVM, which lead to sparser SVM solutions that have better generalization performance.

6.1.1 Leveraging the Duality Gap

One question regarding the optimization scheme in LASVM is the rate at which to perform REPROCESS operations. A simple approach would be to perform one REPROCESS operation after each PROCESS step. However, this heuristic approach may result in under optimization of the objective function in the intermediate steps if this rate is smaller than the optimal proportion. Another option would be to run REPROCESS until a small predefined threshold ε exceeds the L_∞ norm of the projection of the gradient $(\partial G(\alpha)/\partial \alpha_i)$. Little work has been done to determine the correct value of the threshold ε. A geometrical argument relates this norm to the position of the support vectors relative to the margins [89]. As a consequence, one usually chooses a relatively small threshold, typically in the range 10^{-4} to 10^{-2}. Using such a small threshold to determine the rate of RE-

PROCESS operations results in many REPROCESS steps after each PROCESS operation. This will not only increase the training time and computational complexity, but can potentially over optimize the objective function at each iteration. Since non-convex iterations work towards suppressing some training instances (outliers), the intermediate learned models should be well enough trained in order to capture the characteristics of the training data but on the other hand, should not be over-optimized since only part of the entire training data is seen at that point in time. Therefore, it is necessary to employ a criteria to determine an accurate rate of REPROCESS operations after each PROCESS. We define this policy as *the minimization of the gap between the primal and the dual* [17].

Optimization of the Duality Gap From the formulations of the primal and dual functions in (2.17) and (2.20) respectively, it can be shown that the optimal values of the primal and dual are same [90]. At any non-optimal point, the primal function is *guaranteed* to lie above the dual curve. In formal terms, let $\hat{\theta}$ and $\hat{\alpha}$ be solutions of problems (2.17) and (2.20), respectively. The strong duality asserts that for any feasible θ and α,

$$G(\alpha) \leq G(\hat{\alpha}) = J(\hat{\theta}) \leq J(\theta) \qquad with \qquad \hat{\theta} = \sum_i \hat{\alpha}_i \Phi(x_i) \tag{6.1}$$

That is, at any time during the optimization, the value of the primal $J(\cdot)$ is higher than the dual $G(\cdot)$. Using the α notation in (2.21) that permits the α's to take on negative values, and the equality $w = \sum_l \alpha_l x_l$, we show that this holds as follows:

$$J(\theta) - G(\alpha) = \frac{1}{2}\|w\|^2 + C\sum_l |1 - y_l(w \cdot x_l + b)|_+ - \sum_l \alpha_l y_l + \frac{1}{2}\|w\|^2$$

$$
\begin{aligned}
&= \|w\|^2 - \sum_l \alpha_l y_l + C \sum_l |1 - y_l(w \cdot x_l + b)|_+ \\
&= w \sum_l \alpha_l x_l - \sum_l \alpha_l y_l + C \sum_l |1 - y_l(w \cdot x_l + b)|_+ \\
&= -\sum_l y_l \alpha_l |1 - y_l(w \cdot x_l + b)|_+ + C \sum_l |1 - y_l(w \cdot x_l + b)|_+ \\
&= \sum_l \underbrace{(C - \alpha_l y_l)}_{\geq 0} \underbrace{|1 - y_l(w \cdot x_l + b)|_+}_{\geq 0} \\
&\geq 0
\end{aligned}
$$

where $C - \alpha_l y_l \geq 0$ is satisfied by the constraint of the dual function (2.21). Then, the SVM solution is obtained when one reaches $\bar{\theta}, \bar{\alpha}$ such that

$$
\varepsilon > J(\bar{\theta}) - G(\bar{\alpha}) \qquad where \qquad \bar{\theta} = \sum_i \bar{\alpha}_i \Phi(x_i) \tag{6.2}
$$

The strong duality in Equation 6.1 then guarantees that $J(\bar{\theta}) < J(\hat{\theta}) + \varepsilon$. Few solvers implement this criterion since it requires the additional calculation of the gap $J(\theta) - G(\alpha)$. We advocate using criterion (6.2) using a threshold value ε that grows sublinearly with the number of examples. Letting ε grow makes the optimization coarser when the number of examples increases. As a consequence, the asymptotic complexity of optimizations in online setting can be smaller than that of the exact optimization.

Most SVM solvers use the dual formulation of the QP problem. However, increasing the dual does not necessarily reduce the duality gap. The dual function follows a nice monotonically increasing pattern at each optimization step, whereas the primal shows significant up and down fluctuations. In order to keep the size of the duality gap in check, before each PROCESS operation we compute the standard deviation of the primal, which we call the *Gap Target* $\hat{\mathbb{G}}$

$$\hat{\mathbb{G}} = \max(0, \sqrt{\sum_{i=1}^{n} h_i^2 - \frac{\left(\sum_{i=1}^{n} h_i\right)^2}{l}}) \tag{6.3}$$

where l is the number of support vectors and $h_i = C_i y_i g_i$. After computing the gap target, we run a PROCESS step and check the new Gap $\mathcal{G}$ between the primal and the dual. After an easy derivation, the gap is computed as

$$\mathcal{G} = -\sum_{i=1}^{n} (\alpha_i g_i + \max(0, C \cdot g_i)) \tag{6.4}$$

We cycle between running REPROCESS and computing the gap $\mathcal{G}$ until the termination criteria $\mathcal{G} \leq \max(C, \hat{\mathbb{G}})$ is reached. That is, we require the duality gap after the REPROCESS operations to be smaller than or equal to initial gap target $\hat{\mathbb{G}}$. After this point, the learner continues with computing the new Gap Target and running PROCESS and REPROCESS operation on the next fresh instance from the unseen example pool.

6.1.2 Building Blocks

The implementation of LASVM-G maintains the following pieces of information as its key building blocks: the coefficients α_i of the current kernel expansion $\mathcal{S}$, the bounds for each α, and the partial derivatives of the instances in $\mathcal{S}$, given as

$$g_k \;=\; \frac{\partial W(\boldsymbol{\alpha})}{\partial \alpha_k} \;=\; y_k - \sum_i \alpha_i K(x_i, x_k) \;=\; y_k - \hat{y}(x_k) \tag{6.5}$$

The kernel expansion here maintains all the training instances in the learner's active set, both the support vectors and the instances with $\alpha = 0$.

Optimization is driven by two kinds of direction searches. The first operation, PROCESS, inserts an instance into the kernel expansion and initializes its α_i and

gradient g_i (Step 1). After computing the step size (Step 2), it performs a direction search (Step 3).

LASVM-G PROCESS(i)

1) $\alpha_i \leftarrow 0, \qquad g_i \leftarrow y_k - \sum_{s \in S} \alpha_s K_{is}$

2) **If** $g_i < 0$ then

$$\lambda = \max\left\{A_i - \alpha_i, \frac{g_i}{K_{ii}}\right\}$$

Else

$$\lambda = \max\left\{B_i - \alpha_i, \frac{g_i}{K_{ii}}\right\}$$

3) $\alpha_i \leftarrow \alpha_i + \lambda$

$g_s \leftarrow g_s - \lambda K_{is} \quad \forall s$ *in kernel expansion*

We set the offset term for kernel expansion b to zero for computational simplicity. As discussed in the SVM section regarding the offset term, disallowing b removes the necessity of satisfying the constraint $\sum_{i \in \mathcal{S}} \alpha_i = 0$, enabling the algorithm to update a single α at a time, both in PROCESS and REPROCESS operations.

LASVM-G REPROCESS()

1) $i \leftarrow \arg\min_{s \in S} g_s$ with $\alpha_s > A_s$

$j \leftarrow \arg\max_{s \in S} g_s$ with $\alpha_s < B_s$

2) Bail out if (i, j) is not a τ-violating pair.

3) **If** $g_i + g_j < 0$ **then** $g \leftarrow g_i \quad k \leftarrow i$

Else $g \leftarrow g_j, \quad k \leftarrow j$

4) **If** $g < 0$ **then**

$$\lambda = \max\left\{A_i - \alpha_i, \frac{g}{K_{ii}}\right\}$$

Else

$$\lambda = \min\left\{B_i - \alpha_i, \frac{g}{K_{ii}}\right\}$$

5) $\alpha_k \leftarrow \alpha_k + \lambda$

$g_s \leftarrow g_s - \lambda K_{is} \quad \forall s$ *in kernel expansion*

The second operation, REPROCESS, searches all of the instances in the kernel expansion and selects the instance with the maximal gradient (Steps 1-3). Once an instance is selected, LASVM-G computes a step size (Step 4) and performs a direction search (Step 5).

Both PROCESS and REPROCESS operate on the instances in the kernel expansion, but neither of them remove any instances from it. A removal step is necessary for improved efficiency because as the learner evolves, the instances that were admitted to the kernel expansion in earlier iterations as support vectors may not serve as support vectors anymore. Keeping such instances in the kernel expansion slows down the optimization steps without serving much benefit to the learner and increases the application's requirement for computational resources. A straightforward approach to address this inefficiency would be to remove all of the instances with $\alpha_i = 0$, namely all non-support vectors. One concern with this approach is that once an instance is removed, it will not be seen by the learner again, and thus, it will no longer be eligible to become a support vector in the later stages of training. It is important to find a balance between maintaining the efficiency of a small sized kernel expansion and not aggressively removing instances from the kernel expansion. Therefore, the cleaning policy needs to preserve the instances that can potentially become SVs at a later stage of training while removing instances that have the lowest possibility of becoming SV's in the future.

CLEAN

n : *number of non-SVs in the kernel expansion.*

m : *maximum number of allowed non-SVs.*

$\vec{v}$: *Array of partial derivatives.*

1) **If** $n < m$ **return**

2) $\vec{v} \leftarrow \vec{v} \cup |g_i|_+, \ \forall i$ with $\alpha_i = 0$

3) Sort the gradients in $\vec{v}$ in ascending order.

$g_{threshold} \leftarrow v[m]$

4) **If** $|g_i|_+ \geq g_{threshold}$ **then** remove $x_i, \ \forall i$ with $\alpha_i = 0$

Our cleaning procedure periodically checks the number of non-SVs in the kernel expansion If the number of non-SVs n is more than the number of instances that is permitted in the expansion m by the algorithm, CLEAN selects the extra non-SV instances with highest gradients for removal. Note that, it is immaterial to distinguish whether an instance has not been an SV for many iterations or it has just become a non-SV. In either case, those examples do not currently contribute to the classifier and are treated equally from a cleaning point of view.

6.1.3 Online Iterations in LASVM-G

LASVM-G exhibits the same learning principle as LASVM, but in a more systematic way. Both algorithms make one pass (one epoch) over the training set. Empirical evidence suggests that a single epoch over the entire training set yields a classifier as good as the SVM solution. Upon initialization, LASVM-G alternates between its PROCESS and REPROCESS steps during the epoch like LASVM, but distributes LASVM's one time FINISHING step to the optimizations performed in each REPROCESS cycle at each iteration and the periodic CLEAN operations.

LASVM-G

1) **Initialization**:

   Set $\boldsymbol{\alpha} \leftarrow \mathbf{0}$

2) **Online Iterations**:

   Pick an example x_i

   Compute Gap Target $\hat{\mathbb{G}}$

   $Threshold \leftarrow \max(C, \hat{\mathbb{G}})$

   Run PROCESS(x_i)

   while Gap $\mathcal{G} > Threshold$

     Run REPROCESS

   end

   Periodically run CLEAN

Another important property of LASVM-G is that it leverages the duality gap to determine the number of REPROCESS steps after each PROCESS (the -G suffix emphasizes this distinction).

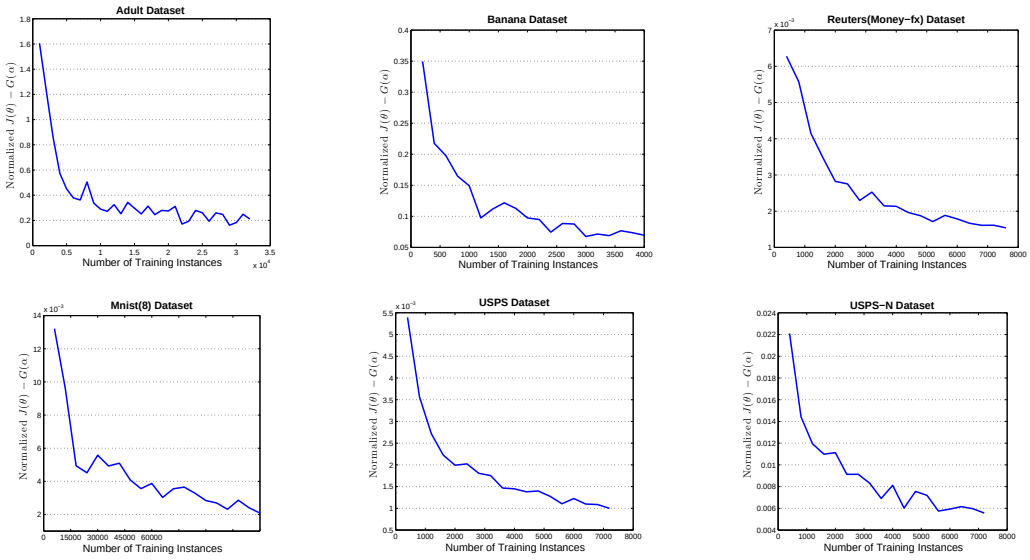
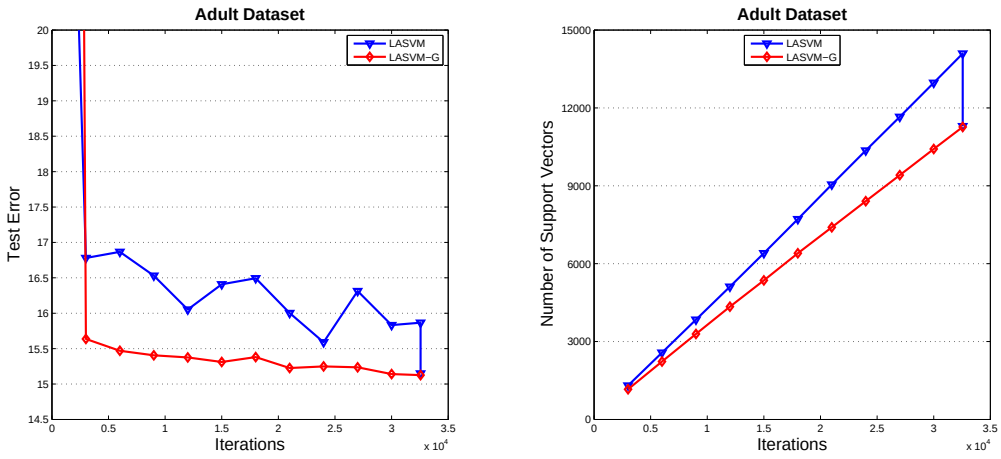


Figure 6.1. The duality gap $(J(\theta) - G(\alpha))$, normalized by the number of training instances. The normalization eliminates the bias on the primal and dual values caused by different number of support vectors at various snapshots of training LASVM-G

Reducing the duality gap too fast can cause over optimization in early stages without yet observing sufficient training data. Conversely, reducing the gap too slow can result in under optimization in the intermediate iterations. Figure 6.1 shows that as the learner sees more training examples, the duality gap gets smaller. The major enhancements that are introduced to LASVM enable LASVM-G to achieve higher prediction accuracies than LASVM in the intermediate stages of training. Figure 6.2 presents a comparative analysis of LASVM-G versus LASVM for the Adult dataset. While both algorithms report the same generalization performance in the end of training, LASVM-G reaches a better classification accuracy at an earlier point in training than LASVM and is able to maintain its performance relatively stable with a more reliable model over the course of training. Furthermore, LASVM-G maintains fewer number of support vectors in the intermediate training steps, as evidenced in Figure 6.2(b).



(a) Test Error Convergence (b) Growth of the Number of Support Vectors

Figure 6.2. Comparison of LASVM and LASVM-G for Adult dataset. We see that LASVM-G arrives at a more accurate SVM solution (Fig. (a)) with fewer support vectors at a faster rate (Fig. (b)). The drop of the Test Error and the number of support vectors in the end of one pass of the iterations for LASVM is the result of the optimizations done by the FINISHING step.

In the next sections, we further introduce three SVM algorithms that are implemented based on LASVM-G, namely LASVM-NC, LASVM-I and FULL SVM. While these SVM algorithms share the main building blocks of LASVM-G, each algorithm exhibits a distinct learning principle. LASVM-NC uses the LASVM-G methodology in a non-convex learner setting. LASVM-I is a learning scheme that we propose as a convex variant of LASVM-NC. FULL SVM does not take advantage of the non-convexity or the efficiency of the CLEAN operation, and acts as a baseline case for comparisons in our experimental evaluation.

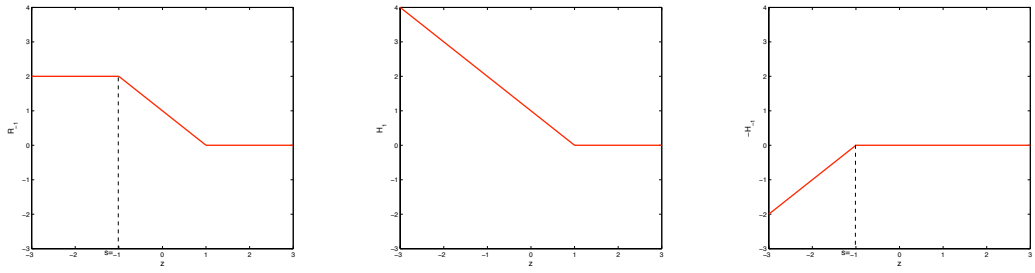
6.2 Non-convex Online SVM – LASVM-NC

In this section, we present LASVM-NC, a non-convex online SVM solver that achieves sparser SVM solutions in less time than online convex SVMs and batch SVM solvers. We first introduce the non-convex Ramp Loss function and discuss how non-convexity can overcome the inefficiencies and scalability problems of convex SVM solvers. We then present the methodology to optimize the non-convex objective function, followed by the description of the online iterations of LASVM-NC.

6.2.1 Ramp Loss

Traditional convex SVM solvers rely on the Hinge Loss H_1 (shown in Figure 6.3(b)) to solve the QP problem, which can be represented in Primal form as

$$\min_{\boldsymbol{w},b} J(\boldsymbol{w},b) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{l=1}^{n} H_1(y_i f(x_i)) \tag{6.6}$$



(a) Ramp Loss (b) Convex Hinge Loss (c) Concave Loss

Figure 6.3. The Ramp Loss 6.3(a) can be decomposed into a Convex Hinge Loss 6.3(b) and a Concave Loss 6.3(c)

In the Hinge Loss formulation $H_s(z) = \max(0, s - z)$, s indicates the Hinge point and the elbow at $s = 1$ indicates the point at which $y_l f_\theta(x_l) = y_l(w \cdot \Phi(x_l) + b) = 1$. Assume for simplicity that the Hinge Loss is made differentiable with a smooth approximation on a small interval $z \in [1 - \epsilon, 1 + \epsilon]$ near the hinge point. Differentiating (6.6) shows that the minimum $\boldsymbol{w}$ must satisfy

$$\boldsymbol{w} = -C \sum_{l=1}^{L} y_l H_1^{'}(y_l) f_\theta(\boldsymbol{x}_l) \Phi(\boldsymbol{x}_i) \tag{6.7}$$

In this setting, correctly classified instances outside of the margin ($z \geq 1$) can not become SVs because $H_1^{'}(z) = 0$. On the other hand, for the training examples with ($z < 1$), $H_1^{'}(z)$ is 1, so they cost a penalty term at the rate of misclassification of those instances. One problem with Hinge Loss based optimization is that it imposes no limit on the influences of the outliers; that is, the misclassification penalty is unbounded. Furthermore in Hinge Loss based optimization, all misclassified training instances become support vectors. Consequently, the number of support vectors scales linearly with the number of training examples [91]. Specifically,

$$\frac{\#SV}{\#Examples} \rightarrow 2\mathfrak{B}_\Phi \tag{6.8}$$

where $\mathfrak{B}_\Phi$ is the best possible error achievable linearly in the feature space $\Phi(\cdot)$. Such fast pace of growth of the number of support vectors becomes prohibitive for training SVMs in large scale datasets.

In practice, not all misclassified training examples are necessarily informative to the learner. For instance in noisy datasets, many instances with label noise become support vectors due to misclassification, even though they are not informative about the correct classification of new instances in recognition. Thus it is reasonable to limit the influence of the outliers and allow the real informative training instances define the model. Many research efforts have been made to deal with class label noise [92, 93, 81, 94, 95, 96], and have suggested that in many situations, eliminating instances that contain class label noise (also known as misclassified instances or outliers) will improve classification accuracy. In SVMs, since Hinge Loss admits all outliers into the SVM solution, we need to select an alternative loss function that enables to selectively ignore the instances that are misclassified according to the current model. For this purpose, we propose to use the Ramp Loss (Figure 6.3(a))

$$R_s(z) = H_1(z) - H_s(z) \tag{6.9}$$

to control the score window for z at which we are willing to convert instances into support vectors. Replacing $H_1(z)$ with $R_s(z)$ in (6.7), we see that the Ramp Loss suppresses the influence of the instances with score $z < s$ by not converting them into support vectors. However, since Ramp Loss is non-convex, it prohibits us from using widely popular optimization schemes devised for convex functions.

While convexity has many advantages and nice mathematical properties, we point out that non-convexity has its own benefits of yielding faster and sparser

solutions. Our aim is to achieve the best of both worlds; generate a reliable and robust SVM solution that is faster and sparser than traditional convex optimizers. This can be achieved by reducing the complexity of non-convex loss function by transforming the problem into a difference of convex parts. We employ the Concave-Convex Procedure (CCCP) [97] to solve the non-convex optimization problem in this fashion. CCCP is closely related to the "Difference of Convex" methods that have been applied to many problems, including dealing with missing values in SVMs [98], improving boosting algorithms [83], and implementing ψ-learning [99, 86]. The elegance of CCCP comes from the fact that it first decomposes a non-convex cost function into a combination of convex parts (by a local approximation of the concave part) and performs optimization on the difference of these convex functions. Formally, CCCP can be described as follows.

Assume that a cost function $J(\boldsymbol{\theta})$ can be decomposed into the sum of a convex part $J_{vex}(\boldsymbol{\theta})$ and a concave part $J_{cav}(\boldsymbol{\theta})$. Each iteration of CCCP approximates the concave part by its tangent and minimizes the resulting convex function.

Algorithm 1 The Concave-Convex Procedure (CCCP)

Initialize $\boldsymbol{\theta}^0$ with a best guess

repeat

$$\boldsymbol{\theta}^{t+1} = \arg\min_{\boldsymbol{\theta}}(J_{vex}(\boldsymbol{\theta}) + J'_{cav}(\boldsymbol{\theta}^t) \cdot \boldsymbol{\theta})$$

until convergence of $\boldsymbol{\theta}^t$

By summing two inequalities for $\boldsymbol{\theta}$ and from the concavity of $J_{cav}(\boldsymbol{\theta})$, it is easy to infer that the cost $J(\boldsymbol{\theta}^t)$ decreases after each iteration:

$$\begin{aligned} J_{vex}(\boldsymbol{\theta^{t+1}}) + J'_{cav}(\boldsymbol{\theta^t}) \cdot \theta^{t+1} &\leq J_{vex}(\boldsymbol{\theta^t}) + J'_{cav}(\boldsymbol{\theta^t}) \cdot \boldsymbol{\theta}^t \\ J_{cav}(\boldsymbol{\theta^{t+1}}) &\leq J_{cav}(\boldsymbol{\theta^t}) + J'_{cav}(\boldsymbol{\theta^t}) \cdot (\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t) \end{aligned} \tag{6.10}$$

We do not need any hyper-parameters for this optimization, and since the problem is now purely convex, we can use any efficient convex algorithm to solve this problem. Similarly, the Ramp Loss can be decomposed into a difference convex parts (as shown in Figure 6.3 and Equation 6.9), which makes it amenable to CCCP optimization. The new cost $J^s(\boldsymbol{\theta})$ after substituting the Hinge Loss with the Ramp Loss then reads:

$$\begin{aligned}\min_{\boldsymbol{\theta}} \; J^s(\boldsymbol{\theta}) &= \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{l=1}^{n} R_s(y_i f(x_i)) \\ &= \underbrace{\frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{l=1}^{n} H_1(y_i f(x_i))}_{J^s_{vex}(\boldsymbol{\theta})} \underbrace{-C\sum_{l=1}^{n} H_s(y_i f(x_i))}_{J^s_{cav}(\boldsymbol{\theta})}\end{aligned} \tag{6.11}$$

For simplification purposes, we introduce the notation

$$\beta_l = y_l \frac{\partial J^s_{cav}(\boldsymbol{\theta})}{\partial f_{\boldsymbol{\theta}}(x_l)} = \begin{cases} C & \text{if } \; y_l f_{\boldsymbol{\theta}}(x_l) < s \\ 0 & \text{otherwise} \end{cases} \tag{6.12}$$

where $f_{\boldsymbol{\theta}}(x_l)$ is the kernel expansion defined as in (2.15) with the offset term $b = 0$. The cost function in Equation 6.11, along with the notation introduced in Equation 6.12 is then reformulated as the following dual optimization problem:

$$\begin{aligned}\max_{\alpha} \; G(\alpha) &= \sum_i y_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j K_{i,j} \\ \text{with} \; &\begin{cases} A_i \le \alpha_i \le B_i \\ A_i = \min(0, Cy_i) - \beta_i y_i \\ B_i = \max(0, Cy_i) - \beta_i y_i \\ \beta_i \;\; \text{from Equation 6.12} \end{cases}\end{aligned} \tag{6.13}$$

Note that disallowing the offset term removes equality constraint for the α's

(which appears in (2.20)), enabling us to update a single α at a time.

There is a fundamental difference between non-convex optimization in batch and online SVMs. Batch non-convex SVMs alternate between solving (6.13) and updating the β's of *all* training instances. LASVM-NC, on the other hand, adjusts the β of only the new fresh instance based on the current model and solves (6.13) while the online algorithm is progressing. We also would like to point out that if the β's of all of the training instances are initialized to zero and left unchanged in the online iterations, the algorithm becomes traditional Hinge Loss SVM. From another viewpoint, if $s \ll 0$, then the β's will remain zero and the effect of Ramp Loss will not be realized. Therefore, (6.13) can be viewed as a generic algorithm that can act as both Hinge Loss SVM and Ramp Loss SVM with CCCP that enables non-convex optimization.

6.2.2 Online Iterations in LASVM-NC

The online iterations in LASVM-NC are similar to LASVM-G in the sense that they are also based on alternating PROCESS and REPROCESS steps, with the distinction of replacing the Hinge Loss with the Ramp Loss. LASVM-NC extends the LASVM-G algorithm with the computation of the $\boldsymbol{\beta}$, followed by updating the α bounds A and B as shown in (6.13). Note that while the $\boldsymbol{\beta}$ do not explicitly appear in the PROCESS and REPROCESS algorithm blocks, they do in fact affect these optimization steps through the new definition of the bounds A and B.

When a new example x_i is encountered, LASVM-NC first computes the β_i for this instance as in the algorithm block, where y_i is the class label, $f_{\boldsymbol{\theta}}(x_i)$ is the decision score for x_i, and s is the score threshold for permitting instances to become support vectors.

It is necessary to initialize the CCCP algorithm appropriately in order to avoid getting trapped in poor local optima. In batch SVMs, this corresponds to running classical SVM on the entire, or on a subset of training instances in the first iteration to initialize CCCP, then it is followed by the non-convex optimization in the subsequent iterations. In the online setting, we initially allow convex optimization for the first few instances by setting their $\beta_i = 0$ (i.e. use Hinge Loss) and then switch to non-convex behavior in the remainder of online iterations.

LASVM-NC

$\mathbb{S}$: *min. number of SVs to start non-convex behavior.*

1) **Initialization**:

Set $\boldsymbol{\beta} \leftarrow \mathbf{0}$, $\boldsymbol{\alpha} \leftarrow \mathbf{0}$

2) **Online Iterations**:

Pick an example x_i

Set $\beta_i = \begin{cases} C & \text{if } \ y_i f_{\boldsymbol{\theta}}(x_i) < s \text{ and } \#\text{SV} > \mathbb{S} \\ 0 & \text{otherwise} \end{cases}$

Set α_i bounds for x_i to

$$(\min(0, Cy_i) - \beta_i y_i \leq \alpha_i \leq \max(0, Cy_i) - \beta_i y_i)$$

Compute Gap Target $\hat{\mathbb{G}}$

$Threshold \leftarrow \max(C, \hat{\mathbb{G}})$

Run PROCESS(x_i)

while Gap $\mathcal{G} > Threshold$

Run REPROCESS

end

Periodically run CLEAN

Note from (6.13) that the α bounds for instances with $\beta = 0$ follow the formulation for the traditional convex setting. On the other hand, the bounds for the instances

Table 6.1. Analysis of Adult dataset in the end of the training of the models. "Admitted" column shows the number of examples that lie on the flat region (left and right) of the Ramp Loss (with $s = -1$) when they were inserted into the expansion. "Cleaned" column shows the number of examples removed during CLEAN.

	Expansion		Admitted		Cleaned	
	# SV	# Non-SV	Ramp(L)	Ramp(R)	Ramp(L)	Ramp(R)
FULL SVM	11831	20731	32562		0	
LASVM-G	11265	0	1340	20252	1	19562

with $\beta = C$, that is, the outliers with score ($z < s$) are assigned new bounds based on the Ramp Loss criteria. Once LASVM-NC establishes the α bounds for the new instance, it computes the Gap Target $\hat{\mathbb{G}}$ and takes a PROCESS step. Then, it makes optimizations of the REPROCESS kind until the size of the duality gap comes down to the Gap Threshold. Finally, LASVM-NC periodically runs CLEAN operation to keep the size of the kernel expansion under control and to maintain its efficiency throughout the training stage.

6.3 LASVM with Ignoring Instances – LASVM-I

This SVM algorithm employs the Ramp function in Figure 6.3(a) as a filter to the learner *prior* to the PROCESS step. That is, once the learner is presented with a new instance, it first checks if the instance is on the ramp region of the function ($1 > y_i \sum_j \alpha_j K_{ij} > s$). The instances that are outside of the ramp region are not eligible to participate in the optimization steps and they are immediately discarded without further action. The rationale is that the instances that lie on the flat regions of the Ramp function will have derivative $H'(z) = 0$, and based on Equation 6.7, these instances will not play role in determining the decision hyperplane $\boldsymbol{w}$.

LASVM-I algorithm is also based on the following recordkeeping that we conducted when running LASVM-G experiments. In LASVM-G, we kept track of two important data points. First, we recorded the position of all instances on the Ramp Loss curve right before inserting the instance into the kernel expansion. Second, we kept track of the number of instances that were removed from the kernel expansion which were on the flat region of the Ramp Loss curve when they were admitted. The numeric breakdown is presented in Table 6.1. Based on the distribution of these cleaned instances, it is evident that most of the cleaned examples that were initially admitted from ($z > 1$) region were removed from the kernel expansion with CLEAN at a later point in time. This is expected, since the instances with ($z > 1$) are already correctly classified by the current model with a certain confidence and hence do not become support vectors.

On the other hand, Table 6.1 shows that almost all of the instances inserted from left flat region (misclassified examples due to $z < s$) became SVs and therefore were never removed from the kernel expansion. Intuitively, the examples that are misclassified by a wide margin should not become support vectors. Ideally, the support vectors should be the instances that are within the margin of the hyperplane. As studies on Active Learning show [3, 43], the most informative instances to determine the hyperplane lie within the margin. Thus, LASVM-I ignores the instances that are misclassified by a margin ($z < s$) up front and prevents them from becoming support vectors.

LASVM-I

1) **Initialization**:

   Set $\boldsymbol{\alpha} \leftarrow \mathbf{0}$

2) **Online Iterations**:

   Pick an example x_i

   Compute $z = y_i \sum_{j=0}^{n} \alpha_j K(x_i, x_j)$

   if ($z > 1$ or $z < s$)

      Skip x_i and bail out

   else

      Compute Gap Target $\hat{\mathbb{G}}$

   $Threshold \leftarrow \max(C, \hat{\mathbb{G}})$

   Run PROCESS(x_i)

   while Gap $\mathcal{G} > Threshold$

     Run REPROCESS

   end

   Periodically run CLEAN

Note that LASVM-I can not be regarded as a non-convex SVM solver since the instances with $\beta = C$ (which corresponds to $z < s$) are already being ignored up front before the optimization steps. Consequently, all the instances visible to the optimization steps have $\beta = 0$, which converts objective function in (6.13) into the convex Hinge Loss from an optimization standpoint. Thus, combining these two filtering criteria ($z > 1$ and $z < s$), LASVM-I trades non-convexity with a filtering Ramp function to determine whether to ignore an instance or proceed with optimization steps. Our goal with designing LASVM-I is that, based on this initial filtering step, it is possible to achieve further speedups in training times while maintaining competitive generalization performance. The experimental results validate this claim.

6.4 LASVM-G without CLEAN – FULL SVM

This algorithm serves as a baseline case for comparisons in our experimental evaluation. The learning principle of FULL SVM is based on alternating between LASVM-G's PROCESS and REPROCESS steps throughout the training iterations. When a new example is encountered, FULL SVM computes the Gap Target (given in Eq. 6.3) and takes a PROCESS step. Then, it makes optimizations of the REPROCESS kind until the size of the duality gap comes down to the Gap Threshold. In this learning scheme, FULL SVM admits every new training example into the kernel expansion without any removal step (i.e. no CLEAN operation). This behavior mimics the behavior of traditional SVM solvers by providing that the learner has constant access to all training instances that it has seen during training and it can make any of them a support vector any time if necessary. The SMO-like optimization in the online iterations of FULL SVM enables it to converge to the batch SVM solution.

FULL SVM

1) **Initialization**:

   Set $\boldsymbol{\alpha} \leftarrow \mathbf{0}$

2) **Online Iterations**:

   Pick an example x_i

   Compute Gap Target $\hat{\mathbb{G}}$

   $Threshold \leftarrow \max(C, \hat{\mathbb{G}})$

   Run PROCESS(x_i)

   while Gap $\mathcal{G} > Threshold$

     Run REPROCESS

   end

Each PROCESS operation introduces a new instance to the learner, updates its α coefficient and optimizes the objective function. This is followed by potentially

	Train Ex.	Test Ex.	# Features	C	$K(x, \bar{x})$
Adult (Census)	32562	16282	122	100	$e^{-0.005\|x-\bar{x}\|^2}$
Banana	4000	1300	2	10	$e^{-\|x-\bar{x}\|^2}$
Mnist (Digit 8)	60000	10000	784	100	$e^{-0.001\|x-\bar{x}\|^2}$
Reuters(Money-fx)	7770	3299	8315	1	$e^{-0.5\|x-\bar{x}\|^2}$
USPS	7329	1969	256	1	$e^{-2\|x-\bar{x}\|^2}$
USPS+N	7329	1969	256	1	$e^{-2\|x-\bar{x}\|^2}$

Table 6.2. Datasets and the train/test splits used in the experimental evaluations. The last two columns show the SVM parameters C and γ for the RBF kernel.

multiple REPROCESS steps, which exploit τ-violating pairs in the kernel expansion. Within each pair, REPROCESS selects the instance with maximal gradient, and potentially can zero the α coefficient of the selected instance. After sufficient iterations, as soon as a τ-approximate solution is reached, the algorithm stops updating the α coefficients. For full convergence to the batch SVM solution, running FULL SVM usually consists of performing a number of epochs where each epoch performs n online iterations by sequentially visiting the randomly shuffled training examples. Empirical evidence suggests that a single epoch yields a classifier almost as good as the SVM solution. For the theoretical explanation of the convergence results of the online iterations, please refer to [2].

The freedom to maintain and access the whole pool of seen examples during training in FULL SVM does come with a price though. The kernel expansion needs to constantly grow as new training instances are introduced to the learner, and it needs to hold all non-SVs in addition to the SVs of the current model. Furthermore, the learner still needs to include those non-SVs in the optimization steps and this additional processing becomes a significant drag on the training time of the learner.

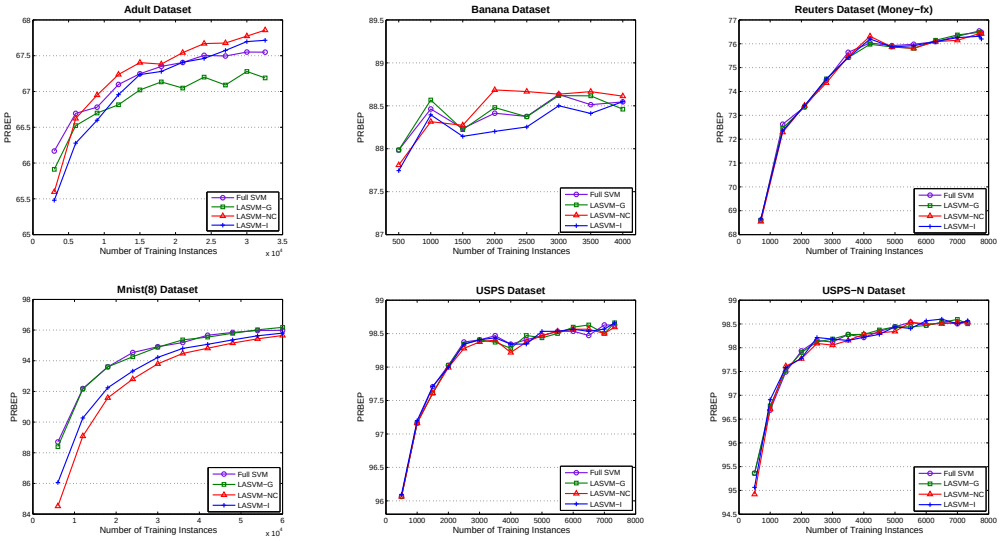


Figure 6.4. Precision/Recall Breakeven Point (PRBEP) vs. Number of Training Instances for all datasets. We used $s = -1$ for the Ramp Loss for LASVM-NC.

6.5 Experiments

The experimental evalutation involves evaluating these outlined SVM algorithms on various datasets in terms of both their classification performances and algorithmic efficiencies leading to scalability. We also compare these algorithms against LIBSVM's available metrics on the same datasets. In the experiments reported below, we run a single epoch over the training examples, all experiments use RBF kernels and the results averaged over 10 runs for each dataset. Table 6.2 presents the characteristics of the datasets and the SVM parameters for running the experiments. In LASVM-G, LASVM-NC and LASVM-I experiments, we empirically set the interval to perform CLEAN at every 300 new training instances.

Generalization Performances One of the metrics that we used in the evaluation of the generalization performances is Precision-Recall Breakeven Point (PRBEP), a widely used metric that measures the accuracy of the positive class where precision equals recall. Figure 6.4 shows the growth of PRBEP curves sam-

Table 6.3. Comparison of all Four SVM algorithms and LIBSVM for all Datasets.

		Datasets					
		Adult	Mnist(8)	Banana	Reuters	USPS	USPSN
Accuracy	FULL SVM	84.87	99.25	90.03	97.19	99.54	98.43
	LASVM-G	84.81	99.27	89.81	97.19	99.54	99.42
	LASVM-NC	85.01	99.15	89.97	97.16	99.52	99.51
	LASVM-I	84.82	99.18	89.84	97.16	99.57	99.49
	LIBSVM	84.91	99.36	90.07	97.19	99.54	99.44
# SV	FULL SVM	11831	3412	947	1122	384	2455
	LASVM-G	11266	3157	941	1120	383	2288
	LASVM-NC	4609	2653	551	1086	372	752
	LASVM-I	5776	2722	669	1093	373	937
	LIBSVM	11420	3146	915	1090	364	2307
Train Time	FULL SVM	1186	4757.4	1	14.4	43.7	36
	LASVM-G	479	547.1	0.6	5.3	17.3	17
	LASVM-NC	129	526.0	0.4	4.7	8	8.3
	LASVM-I	92	491.4	0.3	3.6	5.8	6
	LIBSVM	318	1372.2	0.5	7.2	6.8	52

pled over the course of training for the datasets. Compared to the baseline case FULL SVM, all algorithms are able to maintain competitive generalization performances in the end of training on all examples. Furthermore, LASVM-NC and LASVM-I actually yield better results on some datasets. This can be attributed to their ability to filter *bad* observations (i.e. noise) from training data. In noisy datasets, most of the noisy instances are misclassified and become support vectors in FULL SVM and LASVM-G due to the Hinge Loss. This increase in the number of support vectors (shown in Figure 6.7) causes the SVM to learn complex classification boundaries that can overfit to noise, which can adversely effect their generalization performances. LASVM-NC and LASVM-I are less sensitive to noise, and they learn simpler models that are able to yield better generalization performances under noisy conditions.

For the evaluation of classification performances, we report three other metrics, namely prediction accuracy (in Table 6.3), and AUC and g-means (in Table 6.4).

Table 6.4. Experimental Results that assess the Generalization Performance and Computational Efficiency of all Four SVM algorithms for all Datasets.

		Datasets					
		Adult	Mnist(8)	Banana	Reuters	USPS	USPSN
PRBEP	FULL SVM	67.55	95.98	88.54	76.48	98.62	98.53
	LASVM-G	67.18	96.18	88.46	76.42	98.66	98.50
	LASVM-NC	67.85	95.64	88.61	76.42	98.59	98.53
	LASVM-I	67.71	95.80	88.54	76.20	98.65	98.56
AUC	FULL SVM	0.897	0.998	0.965	0.987	0.999	0.998
	LASVM-G	0.893	0.998	0.966	0.987	0.999	0.998
	LASVM-NC	0.901	0.998	0.965	0.987	0.999	0.998
	LASVM-I	0.899	0.998	0.964	0.987	0.999	0.998
Gmeans	FULL SVM	73.13	97.29	89.51	81.00	98.93	98.42
	LASVM-G	72.87	97.42	89.30	81.17	98.92	98.42
	LASVM-NC	75.03	96.86	89.47	81.19	98.99	98.74
	LASVM-I	73.72	97.06	89.38	81.06	98.89	98.72
# Kernel ($\times 10^6$)	FULL SVM	709.0	2269.7	8.51	33.2	27	30.2
	LASVM-G	233.0	182.6	3.8	9.9	5.2	14.3
	LASVM-NC	116.9	153.5	3.1	9.9	5.2	7.3
	LASVM-I	105.9	121.2	2.0	8	3.2	6.8

We report that all LASVM algorithms yield as good results for these performance metrics as FULL SVM and comparable classification accuracy to LIBSVM. Further, as is the case for PRBEP, LASVM-NC and LASVM-I achieve better results on these metrics for some datasets than FULL SVM and LASVM-G.

We study the impact of the s parameter on the generalization performances of LASVM-NC and LASVM-I and present our findings in Figure 6.5. Since FULL SVM, LASVM-G and LIBSVM do not use Ramp Loss, they are represented with their testing errors and total number of support vectors achieved in the end of training. The Banana dataset shows a clean separation of LASVM-NC and LASVM-I plots, with LASVM-NC curve under the LASVM-I curve. This indicates that LASVM-NC achieves higher classification accuracy with fewer support vectors for all s values for this dataset. In all datasets, increasing the value of s into the

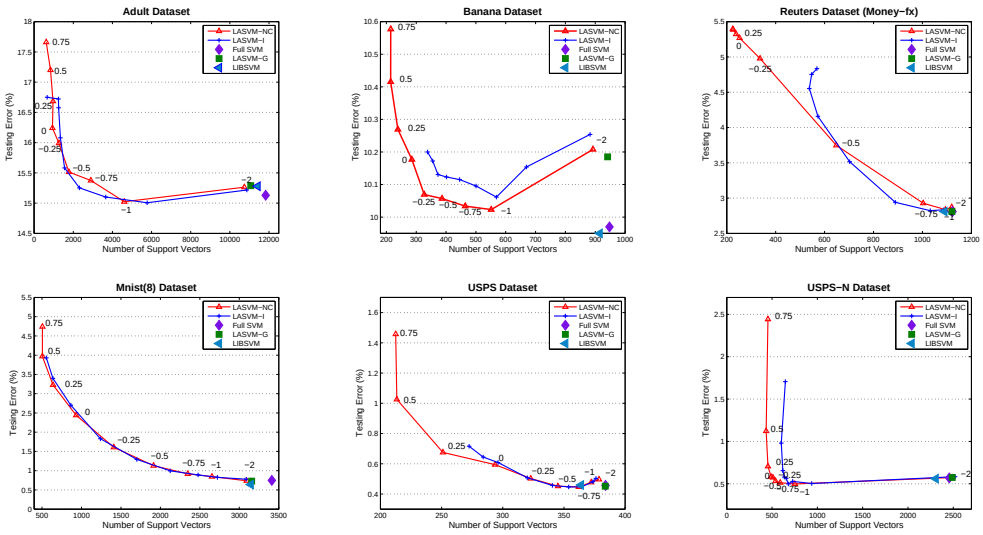


Figure 6.5. Testing Error vs. Number of Support Vectors for various settings of the s parameter of the Ramp Loss.

positive territory actually has the effect of preventing correctly classified instances that are within the margin from becoming SVs. This becomes detrimental to the generalization performance of LASVM-NC and LASVM-I since those instances are among the most informative instances to the learner. Likewise, moving s into further down to the negative territory diminishes the effect of the Ramp Loss on the outliers. If $s \to -\infty$, then $R_s \to H_1$; in other words, if s takes large negative values, the Ramp Loss will not help to remove outliers from the SVM kernel expansion.

It is important to note that at the point $s = -1$, the algorithm behaves as an *Active Learning* framework. As we discussed in Chapter 4, Active Learning is widely known as a querying technique for selecting the most informative instances from a pool of unlabeled instances to acquire their labels. Even in cases where the labels for all training instances are available beforehand, active learning can still be leveraged to select the most informative instances from training sets. We showed in Chapter 4 that querying for the most informative example does not

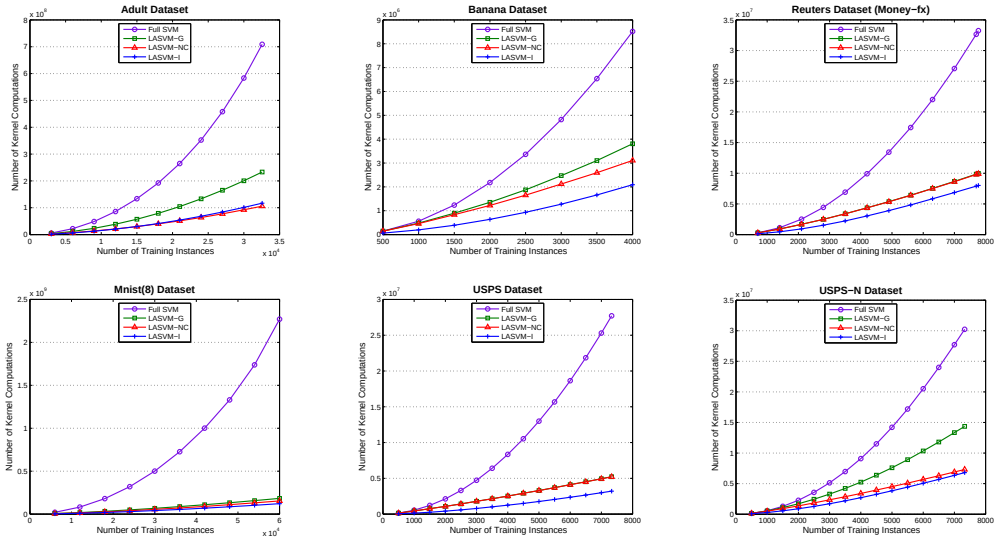


Figure 6.6. Number of Kernel Computations vs. Number of Training Instances

need to be done from the entire training set, but instead, querying from randomly picked small pools can work equally well in a more efficient way [3]. *Small pool active learning* first samples M random training examples from the entire training set and selects the best one among those M examples (see Section 4.2.1). In the extreme case of small pool active learning, setting the size of the pool to 1 corresponds to investigating whether that instance is within the margin or not. In this regard, setting $s = -1$ for the Ramp Loss in LASVM-NC and LASVM-I constrains the learner's focus only on the instances within the margin. Empirical evidence suggests that LASVM-NC and LASVM-I algorithms exhibit the benefits of active learning at $s = -1$ point, which seems to yield optimal results in most of our experiments. However, the exact setting for the s hyperparameter should be determined by the requirements of the classification task and the characteristics of the dataset.

Computational Efficiency A significant time consuming operation of SVMs is the computation of kernel products $K(i,j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. For each new

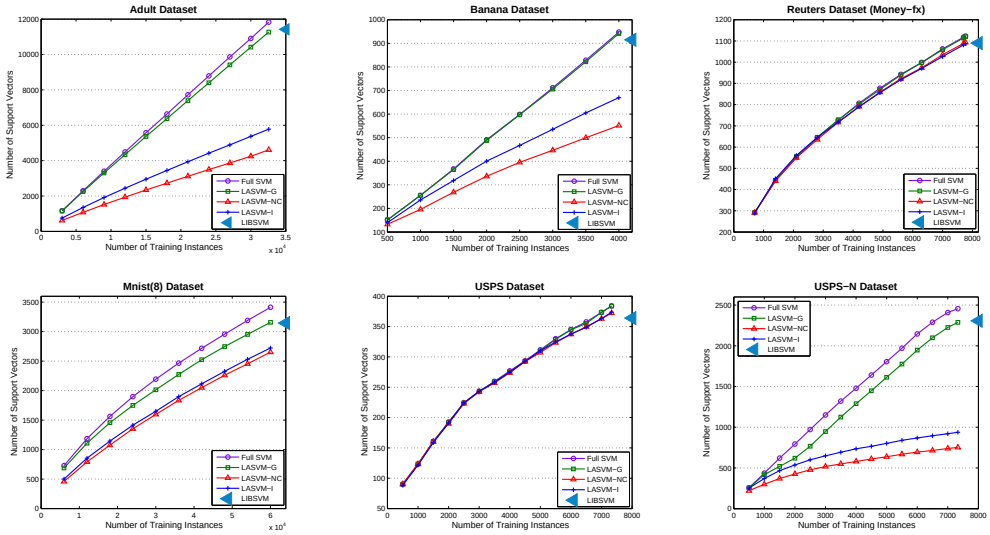


Figure 6.7. Number of Support Vectors vs. Number of Training Instances

example, its kernel product with every instance in the kernel expansion needs to be computed. By reducing the number of kernel computations, it is possible to achieve significant computational efficiency improvements over traditional SVM solvers. In Figure 6.6, we report the number of kernel calculations performed over the course of training iterations. FULL SVM suffers from uncontrolled growth of the kernel expansion, which results in steep increase of the number of kernel products. This also shows why SVMs can not handle large scale datasets efficiently. In comparison, LASVM-G requires fewer kernel products than FULL SVM since LASVM-G keeps the number of instances in the kernel expansion under control by periodically removing uninformative instances through CLEAN operations.

LASVM-NC and LASVM-I yield significant reduction in the number of kernel computations and their benefit is most pronounced in the noisy datasets, Adult, Banana and USPS-N. LASVM-I achieves better reduction of kernel computations than LASVM-NC. This is due to the aggressive filtering done in LASVM-I where no kernel computation is performed for the instances on the flat regions of the Ramp

Loss. On the other hand, LASVM-NC admits those instances into the kernel expansion but achieves sparsity through the non-convex optimization steps. The reason for the low number of kernel products in LASVM-NC is due to its ability to create sparser models than other three algorithms. A comparison of the growth of the number of support vectors during the course of training is shown in Figure 6.7. LASVM-NC and LASVM-I end up with smaller number of support vectors than FULL SVM, LASVM-G and LIBSVM. Furthermore, compared to LASVM-I, LASVM-NC builds noticeably sparser models with less support vectors in noisy Adult, Banana and USPS-N datasets. LASVM-I, on the other hand, makes fewer kernel calculations than LASVM-NC for those datasets. This is a key distinction of these two algorithms: The computational efficiency of LASVM-NC is the result of its ability to build sparse models. Conversely, LASVM-I creates comparably more support vectors than LASVM-NC, but makes fewer kernel calculations due to early filtering. The overall training times for all datasets and all algorithms are presented both in Figure 6.8 and Table 6.4. All three LASVM algorithms are significantly more efficient than FULL SVM. LASVM-NC and LASVM-I also yield faster training than LIBSVM. The fastest training times belong to LASVM-I where LASVM-NC comes close second. The sparsest solutions are achieved by LASVM-NC and this time LASVM-I comes close second. These two algorithms represent a compromise between training time versus sparsity and recognition time, and the appropriate algorithm should be chosen based on the requirements of the classification task.

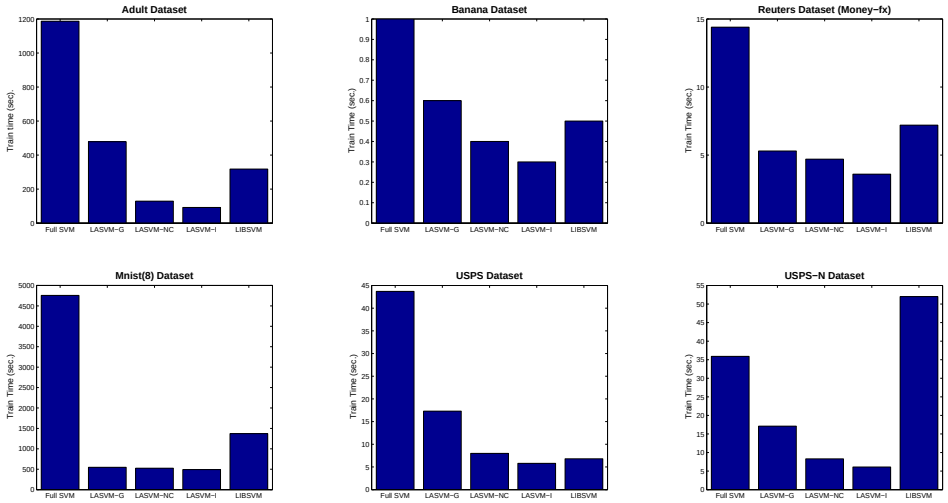


Figure 6.8. Training times of the algorithms for all datasets after one pass over the training instances. The speed improvement in training time becomes more evident in larger datasets.

6.6 Remarks

In traditional convex SVM optimization, the number of support vectors scales linearly with the number of training examples, which unreasonably increases the training time and computational resource requirements. This fact has hindered widespread adoption of SVMs for classification tasks in large-scale datasets. In this chapter, we have studied the ways in which the computational efficiency of an online SVM solver can be improved without sacrificing the generalization performance. This work is concerned with suppressing the influences of the outliers, which particularly becomes problematic in noisy data classification. For this purpose, we first present a systematic optimization approach for an online learning framework to generate more reliable and trustworthy learning models in intermediate iterations (LASVM-G). We then propose two online algorithms, LASVM-NC and LASVM-I, which leverage the Ramp function to avoid the outliers to become support vectors. LASVM-NC replaces the traditional Hinge Loss with the Ramp

Loss and brings the benefits of non-convex optimization using CCCP to an online learning setting. LASVM-I uses the Ramp function as a filtering mechanism to discard the outliers during online iterations. Empirical evidence suggests that the algorithms provide efficient and scalable learning with noisy datasets in two respects: *i*) *computational*: there is a significant decrease in the number of computations and running time during training and recognition, and *ii*) *statistical*: there is a significant decrease in the number of examples required for good generalization. Our findings also reveal that discarding the outliers by leveraging the Ramp function is closely related to the working principles of margin based Active Learning.

Chapter 7

Conclusions

"İlim ilim bilmektir
İlim kendin bilmektir
Sen kendini bilmezsen
Ya nice okumaktır"

Yunus Emre

The goal of machine learning is to turn data into information based on past experience and build decision systems that can act on that information. This goal has attracted interest from various domains, and today, machine learning solutions have become indispensable tools in many fields of science, business and engineering. Along with their benefits, we have also noted issues related to the scalability and stability of machine learning algorithms. These issues can also be characterized as stemming from the *quantity*, *quality* and the *distribution* of the data. Regarding the quantity aspect, we are producing data at a faster rate than before, and we need efficient algorithms that can respond to the requirements of learning from large scale datasets. These requirements include obtaining labels of training examples and reaching out to the most informative instances in the train-

ing data in a cost efficient way, training models in reasonable time, and building simpler models that use less memory in training and recognition phases. The concern about the data quality generally stems from noise in the input data, which degrades the generalization performance and computational efficiency of learning algorithms. The data distribution aspect is concerned with significantly uneven number of instances for classes, which prevents the learners to identify the target class instances in the recognition phase.

This thesis presents methodologies that address these aforementioned issues with the goal of improving scalability, computational and data efficiency and generalization performance of machine learning algorithms in the context of online and active learning with particular focus on Support Vector Machines. Online learning algorithms are usually associated with problems where the complete training set is not available beforehand and active learning is mostly regarded as a technique that addresses unlabeled training instance problem. However, if we look at these learning principles from a broader perspective, it becomes evident that (i) large scale learning problems can substantially benefit from the computational properties of online learning, even if the entire dataset is available beforehand, and (ii) even if the labels of all training instances are available, principles of active learning can be immensely beneficial to the scalability of machine learning algorithms by obtaining the informative instances in training sets and dismissing the redundant and noisy data. Learning with active sample selection is better suited to the incremental working principle of online learning algorithms, since the new informative instance selected by active learning can be integrated to the existing model without retraining all the samples repeatedly.

The common and easy way to deal with very large scale data in learning problems is to randomly sample the entire data and build a model using a subset of

the entire available corpus. This "passive" sampling scheme is often employed to present the learning algorithm a smaller view of the entire dataset that can be handled within time and computational resource (i.e. memory) constraints. Active learning is selective sampling, where the sampling is primarily driven by the queries of the learner to find the informative instances that will have the most impact on the generalization performance of the learner. Thus, instead of focusing on an arbitrary subset of the dataset, the active learner intelligently guides the sampling process to constrain its focus on the instances which best represent the concept that the algorithm is trying to learn. Active learning therefore enables to reach competitive generalization accuracies with less data, yielding fast and data efficient learning. Furthermore, even in the absence of limitations on computing resources and time, active learning can still be used for its generalization behavior. Our observations reveal that active sampling strategy can achieve even higher generalization performance with less data than one can achieve by training on the entire dataset.

In this thesis, we first present an online SVM algorithm, namely LASVM, that outspeeds batch SVM solvers while yielding competitive misclassification rates after a single pass over the training examples. We then show how active example selection in LASVM leads to faster training, higher accuracies, and simpler models, using only a fraction of the training example labels. We show that active learning addresses the class imbalance problem by providing the learner with a more balanced data distribution. We introduce *small pools active learning* method to make the application of active learning to very large datasets computationally feasible. Combined with the early stopping heuristics, we present that our active example selection scheme achieves a fast and scalable solution without sacrificing, and in some cases yielding better generalization performance. We further propose

VIRTUAL, an active learning based adaptive oversampling methodology for imbalanced data classification. Unlike conventional oversampling techniques that create virtual instances for each instance of the minority class, VIRTUAL creates new instances for the support vectors of the minority class. Since VIRTUAL focuses on the most informative instances, it creates fewer synthetic instances and it is computationally more efficient than other oversampling methods, yielding superior prediction performance. Another contribution of this thesis is a non-convex online support vector machine algorithm. First, we introduce LASVM-G that reorganizes the number of REPROCESS operations to develop more trustworthy intermediate models in online iterations by leveraging the duality gap. We then present LASVM-NC, a non-convex online SVM solver that has strong ability of suppressing the influence of outliers. Then, again in the online learning setting, we propose an outlier filtering mechanism, LASVM-I, based on approximating non-convex behavior in convex optimization. Compared to other state of the art SVM solvers, we show that both algorithms are capable of significantly reducing model complexity and training time, especially in noisy data classification.

The challenges in supervised learning algorithms are like the faces in Rubik's Cube. When the focus is on improving the generalization performance of the algorithms, training times may become unreasonably long. On the other hand, when the algorithms are designed and modified to be faster, the prediction performance might be sacrificed. Likewise, training a learner using a large scale dataset possibly takes a long training time and probably yields small classification error. Conversely, a randomly selected small dataset is likely to take shorter time to train at the expense of increased classification error. Analogically, when we try to solve one face of the Rubic's cube, we mix the colors on other faces. It is therefore essential to *simultaneously* address the issues of accuracy, computational efficiency,

data efficiency and scalability. The algorithms proposed in this thesis realize accuracy, efficiency and scalability in three respects: (*i*) *computational performance*: there is a significant decrease in the number of computations and running time during training and recognition, (*ii*) *statistical performance*: there is a significant decrease in the number of examples required for good generalization, and (*iii*) *generalization performance*: the algorithms yield competitive or even better prediction performance in classification tasks.

Faced with the need to analyze the ever growing amount of data, one of the main goals of computing researchers should be to design and develop approaches, algorithms and procedures that are fast, accurate, robust and scalable. In my dissertation, I aimed to reach that goal.

7.1 Future Research Directions

We identify several future research directions that we think is intriguing to investigate.

There has been a surge in the interest for mining data streams from numerous emerging applications, including network traffic monitoring, environmental sensor networks, web click stream mining, social network analysis and financial fraud detection. Unlike traditional data mining, which extracts models and patterns from large amounts of information stored in data repositories, data stream mining is concerned with extracting knowledge structures represented in models and patterns in non-stopping streams of information. This paradigm shift demands a new set of data mining and machine learning techniques that can answer the challenges in mining data streams and the temporal characteristics of the data. The challenges range from handling changes to the statistical properties of the observed data

to changes to the actual set of features that represent the data. From a machine learning perspective, this is known as *concept drift*, a paradigm that adversely effect the generalization performance of learning algorithms. We think it is necessary to study how online learning systems can handle data streams with concept drifts.

Online decision making under uncertainty and time constraints represents a challenging problem in machine learning. Our work is geared towards solving deterministic optimization problems. That is, we do not observe any uncertainty in the underlying data distribution and future observations conform to the characteristics of the instances that were used to build the learned model. Incorporating uncertainty through probabilistic constraints or perturbations to input data requires non-deterministic solutions that can work in an adaptive online setting. These algorithms have far reaching applications in online optimization problems in scheduling, routing, and resource allocation.

It is possible to further improve the effectiveness of active learning in real world settings by taking into account the effect of certain costs. One of the main issues involved with supervised learning is that labeled data typically is obtained at a higher cost than unlabeled data due to the fact that manual labeling of observations takes time, requires human workforce and has financial costs. Traditional active learning aims to minimize misclassification rates without considering such costs associated with obtaining class labels. If improving the classification accuracy by some fraction is not justified by the required cost, then it makes more sense to trade less accuracy for cost efficiency, but the learner can not readily make that judgement. One possible way to extend active learning would be to explore the potential of analyzing active learning in a decision theoretic framework. That is, given the cost of probing for the class label of an instance in the collection, a decision theoretic active learner can look at the problem from a global perspective,

and then decide if it's globally optimal to learn the class label of a particular observation, both in terms of the increase in classification accuracy and the associated costs with acquiring the label. Such a system potentially will have a tremendous benefit and enable us to realize the full potential of active learners in real world settings.

Bibliography

[1] Cherkassky, V. S. and F. Mulier (2007) *Learning from Data: Concepts, Theory, and Methods*, John Wiley & Sons, Inc., New York, NY, USA.

[2] Bordes, A., S. Ertekin, J. Weston, and L. Bottou (2005) "Fast Kernel Classifiers with Online and Active Learning," *Journal of Machine Learning Research (JMLR)*, **6**, pp. 1579–1619.

[3] Ertekin, S., J. Huang, L. Bottou, and L. Giles (2007) "Learning on the border: active learning in imbalanced data classification," in *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, ACM, New York, NY, USA, pp. 127–136.

[4] Ertekin, S., J. Huang, and C. L. Giles, "Adaptive Resampling with Active Learning," Under Review.

[5] Ertekin, S., L. Bottou, and C. L. Giles, "Ignorance is Bliss: Non-convex Online Support Vector Machines," Under Review.

[6] Vapnik, V. (1995) *The Nature of Statistical Learning Theory*, Springer, New York.

[7] Karush, W. (1939) *Minima of Functions of Several Variables with Inequalities as Side Conditions*, Master's thesis, Department of Mathematics, University of Chicago, Chicago, IL, USA.

[8] Kuhn, H. W. and A. W. Tucker (1951) "Nonlinear Programming," in *Proceedings of Second Berkeley Symposium on Mathematical Statistics and Probability.*, University of California Press, pp. 481–492.

[9] Fletcher, R. (1987) *Practical methods of optimization; (2nd ed.)*, Wiley-Interscience, New York, NY, USA.

[10] Cortes, C. and V. Vapnik (1995) "Support-Vector Networks," *Machine Learning*, **20**(3), pp. 273–297.

[11] Vapnik, V. N. (1982) *Estimation of Dependences Based on Empirical Data*, Springer Series in Statistics, Springer-Verlag, Berlin.

[12] Platt, J. (1999) "Fast Training of Support Vector Machines using Sequential Minimal Optimization," in *Advances in Kernel Methods — Support Vector Learning* (B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds.), MIT Press, Cambridge, MA, pp. 185–208.

[13] Chang, C.-C. and C.-J. Lin. (2001), "LIBSVM : a library for support vector machines." .

[14] Collobert, R. and S. Bengio (2001) "SVMTorch: Support Vector Machines for Large-Scale Regression Problems," *Journal of Machine Learning Research*, **1**, pp. 143–160.

[15] Vapnik, V. N., T. G. Glaskova, V. A. Koscheev, A. I. Mikhailski, and A. Y. Chervonenkis (1984) *Algorithms and Programs for Dependency Estimation,* in Russian, Nauka, Moscow.

[16] Keerthi, S. S. and E. G. Gilbert (2002) "Convergence of a Generalized SMO Algorithm for SVM Classifier Design," *Mach. Learn.*, **46**(1-3), pp. 351–360.

[17] Schölkopf, B. and A. J. Smola (2002) *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, USA.

[18] Shalev-Shwartz, S. and N. Srebro (2008) "SVM optimization: inverse dependence on training set size," in *ICML '08: Proceedings of the 25th international conference on Machine learning*, ACM, New York, NY, USA, pp. 928–935.

[19] Steinwart, I. (2004) "Sparseness of Support Vector Machines—Some Asymptotically Sharp Bounds," in *Advances in Neural Information Processing Systems 16* (S. Thrun, L. K. Saul, and B. Schölkopf, eds.), MIT Press, Cambridge, MA.

[20] Joachims, T. (1999) "Making large-scale support vector machine learning practical," , pp. 169–184.

[21] Aizerman, A., E. M. Braverman, and L. I. Rozoner (1964) "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control*, **25**, pp. 821–837.

[22] Rosenblatt, F. (1958) "The Perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, **65**, pp. 386–408.

[23] Novikoff, A. B. J. (1962) "On convergence proofs on perceptrons," in *Proceedings of the Symposium on the Mathematical Theory of Automata*, vol. 12, Polytechnic Institute of Brooklyn, pp. 615–622.

[24] Freund, Y. and R. E. Schapire (1999) "Large Margin Classification Using the Perceptron Algorithm," *Machine Learning*, **37**(3), pp. 277–296.

[25] Cristianini, N. and J. Shawe-Taylor (2000) *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press.

[26] Friess, T.-T., N. Cristianini, and C. Campbell (1998) "The Kernel-Adatron Algorithm: A Fast and Simple Learning Procedure for Support Vector Machines," in *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 188–196.

[27] Gentile, C. (2002) "A new approximate maximal margin classification algorithm," *Journal of Machine Learning Research*, **2**, pp. 213–242.

[28] Li, Y. and P. M. Long (2002) "The Relaxed Online Maximum Margin Algorithm," *Machine Learning*, **46**(1–3), pp. 361–387.

[29] Crammer, K. and Y. Singer (2003) "Ultraconservative online algorithms for multiclass problems," *Journal of Machine Learning Research*, **3**, pp. 951–991.

[30] Crammer, K., J. Kandola, and Y. Singer (2004) "Online Classification on a Budget," in *Advances in Neural Information Processing Systems 16* (S. Thrun, L. Saul, and B. Schölkopf, eds.), MIT Press, Cambridge, MA.

[31] Weston, J., A. Bordes, and L. Bottou (2005) "Online (and Offline) on an Even Tighter Budget," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTAT'05)* (R. G. Cowell and Z. Ghahramani, eds.), Society for Artificial Intelligence and Statistics, pp. 413–420.

[32] Bottou, L., C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. Le Cun, U. A. Muller, E. Säckinger, P. Simard, and V. Vapnik (1994) "Comparison of classifier methods: a case study in handwritten digit recognition," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Conference B: Computer Vision & Image Processing.*, vol. 2, IEEE, Jerusalem, pp. 77–82.

[33] Schölkopf, B. and A. J. Smola (2002) *Learning with Kernels*, MIT Press, Cambridge, MA.

[34] Graf, H. P., E. Cosatto, L. Bottou, I. Durdanovic, and V. Vapnik (2004) "Parallel Support Vector Machines: The Cascade SVM," in *Neural Information Processing Systems*.

[35] Collobert, R., S. Bengio, and Y. Bengio (2002) "A parallel mixture of SVMs for very large scale problems," *Neural Computation*, **14**(5), pp. 1105–1114.

[36] Dasgupta, S. (2005) "Coarse sample complexity bounds for active learning," in *Advances in Neural Information Processing Systems 18*, Vancouver, Canada.

[37] Balcan, M.-F., A. Beygelzimer, and J. Langford (2006) "Agnostic active learning," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, ACM, New York, NY, USA, pp. 65–72.

[38] Cohn, D. A., Z. Ghahramani, and M. I. Jordan (1996) "Active Learning with Statistical Models," *Journal of Artificial Intelligence Research*, **4**, pp. 129–145.

[39] Roy, N. and A. McCallum (2001) "Toward Optimal Active Learning through Sampling Estimation of Error Reduction," in *ICML '01: Proceedings of the 18th International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 441–448.

[40] Freund, Y., H. S. Seung, E. Shamir, and N. Tishby (1997) "Selective Sampling Using the Query by Committee Algorithm," *Machine Learning*, **28**(2-3), pp. 133–168.

[41] Lewis, D. D. and J. Catlett (1994) "Heterogeneous uncertainty sampling for supervised learning," in *In Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann, pp. 148–156.

[42] Tong, S. and D. Koller (2001) "Support vector machine active learning with applications to text classification," *Journal of Machine Learning Research*, **2**, pp. 45–66.

[43] Schohn, G. and D. Cohn (2000) "Less is More: Active Learning with Support Vector Machines," in *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 839–846.

[44] Campbell, C., N. Cristianini, and A. J. Smola (2000) "Query Learning with Large Margin Classifiers," in *ICML '00: Proceedings of the Seventeenth*

International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 111–118.

[45] Atlas, L., D. Cohn, R. Ladner, M. A. El-Sharkawi, and R. J. Marks, II (1990) "Training connectionist networks with queries and selective sampling," , pp. 566–573.

[46] Fedorov, V. V. (1972), "Theory of Optimal experiments." .

[47] Eisenberg, B. and R. L. Rivest (1990) "On the sample complexity of pac-learning using random and chosen examples," in *COLT '90: Proceedings of the third annual workshop on Computational learning theory*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 154–162.

[48] Domingo, C. and O. Watanabe (2000) "MadaBoost: A Modification of AdaBoost," in *COLT '00: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 180–189.

[49] Vishwanathan, S. V. N., A. J. Smola, and M. N. Murty (2003) "SimpleSVM," in *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pp. 760–767.

[50] Tsang, I. W., J. T. Kwok, and C. W. Bay (2005) "Very large SVM training using core vector machines," in *Proceedings 10th International Workshop Artif. Intell. Stat*, pp. 349–356.

[51] Huang, J., S. Ertekin, and C. L. Giles (2006) "Efficient Name Disambiguation for Large Scale Datasets," in *Proceedings of European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, vol. 4213/2006, pp. 536–544.

[52] Bilenko, M., R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg (2003) "Adaptive Name Matching in Information Integration," *IEEE Intelligent Systems*, **18**(5), pp. 16–23.

[53] Ester, M., H.-P. Kriegel, J. Sander, and X. Xu (1996) "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with

Noise," in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 226–231.

[54] Han, H., C. L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis (2004) "Two supervised learning approaches for name disambiguation in author citations," in *Proceedings of Joint Conference on Digital Libraries(JCDL 2004)*, pp. 296–305.

[55] Han, H., H. Zha, and C. L. Giles (2005) "Name disambiguation in author citations using a K-way spectral clustering method," in *Proceedings of Joint Conference on Digital Libraries(JCDL 2005)*, pp. 334–343.

[56] MacKay, D. J. C. (1992) "Information-based objective functions for active data selection," *Neural Comput.*, **4**(4), pp. 589–603.

[57] Bakir, G. H., L. Bottou, and J. Weston (2005) "Breaking SVM Complexity with Cross-Training," in *Advances in Neural Information Processing Systems 17* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), MIT Press, Cambridge, MA, pp. 81–88.

[58] Grzymala-Busse, J. W., Z. Zheng, L. K. Goodwin, and W. J. Grzymala-Busse (2000) "An Approach to Imbalanced Datasets Based on Changing Rule Strength." in *Proceedings of AAAI Workshop on Learning from Imbalanced Datasets*, pp. 69–74.

[59] Chan, P. K. and S. J. Stolfo (1998) "Toward Scalable Learning with Non-Uniform Class and Cost Distributions: A Case Study in Credit Card Fraud Detection," in *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 164–168.

[60] Dumais, S., J. Platt, D. Heckerman, and M. Sahami (1998) "Inductive Learning Algorithms and Representations for Text Categorization," in *Proceedings of International Conference on Information and Knowledge Management (CIKM)*, ACM, New York, NY, USA, pp. 148–155.

[61] Radivojac, P., N. V. Chawla, A. K. Dunker, and Z. Obradovic (2004) "Classification and Knowledge Discovery in Protein Databases," *Journal of Biomedical Informatics*, **37**(4), pp. 224–239.

[62] Kubat, M., R. C. Holte, and S. Matwin (1998) "Machine Learning for the Detection of Oil Spills in Satellite Radar Images," *Machine Learning*, **30**(2-3), pp. 195–215.

[63] Provost, F. (2000) "Machine Learning from Imbalanced Datasets 101," in *Proceedings of AAAI Workshop on Imbalanced Data Sets*.

[64] Abe, N. (2003) "Invited talk: Sampling Approaches to Learning from Imbalanced Datasets: Active Learning, Cost Sensitive Learning and Beyond," *Proceedings of ICML Workshop: Learning from Imbalanced Data Sets*.

[65] Domingos, P. (1999) "MetaCost: A General Method for Making Classifiers Cost-Sensitive," in *Proceedings of International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, New York, NY, USA, pp. 155–164.

[66] Pazzani, M., C. Merz, P. Murphy, K. Ali, T. Hume, and C. Brunk (1994) "Reducing Misclassification Costs," in *Proceedings of 11th International Conference on Machine Learning (ICML)*.

[67] Chawla, N. V., K. W. Bowyer., L. O. Hall, and W. P. Kegelmeyer (2002) "SMOTE: Synthetic Minority Over-sampling Technique." *Journal of Artificial Intelligence Research (JAIR)*, **16**, pp. 321–357.

[68] Japkowicz, N. (2000) "The Class Imbalance Problem: Significance and Strategies," in *Proceedings of International Conference on Artificial Intelligence (IC-AI'2000)*, vol. 1, pp. 111–117.

[69] Kubat, M. and S. Matwin (1997) "Addressing the Curse of Imbalanced Training Datasets: One Sided Selection," *Proceedings of International Conference on Machine Learning (ICML)*, **30**(2-3), pp. 195–215.

[70] Ling, C. X. and C. Li (1998) "Data Mining for Direct Marketing: Problems and Solutions," in *Proceedings of International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 73–79.

[71] Liu, X.-Y., J. Wu, and Z.-H. Zhou (2006) "Exploratory Under-Sampling for Class-Imbalance Learning," in *Proceedings of the International Conference*

on Data Mining (ICDM), IEEE Computer Society, Washington, DC, USA, pp. 965–969.

[72] Japkowicz, N. (1995) "A Novelty Detection Approach to Classification," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 518–523.

[73] Raskutti, B. and A. Kowalczyk (2004) "Extreme re-balancing for SVMs: a case study," *SIGKDD Explorations Newsletter*, **6**(1), pp. 60–69.

[74] Akbani, R., S. Kwek, and N. Japkowicz (2004) "Applying Support Vector Machines to Imbalanced Datasets," in *Proceedings of European Conference on Machine Learning (ECML)*, pp. 39–50.

[75] Smola, A. J. and B. Schölkopf (2000) "Sparse Greedy Matrix Approximation for Machine Learning," in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 911–918.

[76] Japkowicz, N. and S. Stephen (2002) "The class imbalance problem: A systematic study." *Intelligent Data Analysis*, **6**(5), pp. 429–449.

[77] Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984) *Classification and Regression Trees*, Wadsworth.

[78] Bousquet, O. and A. Elisseeff (2002) "Stability and generalization," *Journal of Machine Learning*, **2**.

[79] Shawe-Taylor, J. and N. Cristianini (2004) *Kernel Methods for Pattern Analysis*, Cambridge University Press.

[80] Zhu, X. and X. Wu (2004) "Class noise vs. attribute noise: a quantitative study of their impacts," *Artificial Intelligence Review*, **22**(3), pp. 177–210.

[81] Brodley, C. E. and M. A. Friedl (1999) "Identifying Mislabeled Training Data," *Journal of Artificial Intelligence Research*, **11**, pp. 131–167.

[82] Mason, L., P. L. Bartlett, and J. Baxter (2000) "Improved Generalization through Explicit Optimization of Margins," *Machine Learning*, **38**, pp. 243–255.

[83] Krause, N. and Y. Singer (2004) "Leveraging the margin more carefully," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, ACM, New York, NY, USA, p. 63.

[84] Perez-Cruz, F., A. Navia-Vazquez, and A. R. Figueiras-Vidal (2002) "Empirical Risk Minimization for Support Vector Classifiers," *IEEE Transactions on Neural Networks*, **14**, pp. 296–303.

[85] Linli Xu, D. S., Koby Cramer (2006) "Robust Support Vector Machine Training via Convex Outlier Ablation," in *Twenty-First National Conference on Artificial Intelligence (AAAI)*.

[86] Liu, Y., X. Shen, and H. Doss (2005) "Multicategory ψ learning and Support Vector Machine: Computational Tools," *Journal of Computational and Graphical Statistics*, **14**, pp. 219–236.

[87] Wang, L., H. Jia, and J. Li (2008) "Training robust support vector machine with smooth Ramp loss in the primal space," *Neurocomputing*, pp. 3020 – 3025.

[88] Collobert, R., F. Sinz, J. Weston, and L. Bottou (2006) "Trading convexity for scalability," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*, ACM, New York, NY, USA, pp. 201–208.

[89] Keerthi, S. S., S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy (2001) "Improvements to Platt's SMO Algorithm for SVM Classifier Design," *Neural Computation*, **13**(3), pp. 637–649.

[90] Chapelle, O. (2007) "Training a Support Vector Machine in the Primal," *Neural Computation*, **19**(5), pp. 1155–1178.

[91] Steinwart, I. (2003) "Sparseness of support vector machines," *Journal of Machine Learninng Research*, **4**, pp. 1071–1105.

[92] John, G. H. (1995) "Robust decision trees: Removing outliers from databases," in *Proceedings of 1st International Conference on Knowledge Discovery and Data Mining*, AAAI Press, pp. 174–179.

[93] Zhao, Q. and T. Nishida (1995) "Using Qualitative Hypotheses to Identify Inaccurate Data," *Journal of Artificial Intelligence Research*, **3**, pp. 119–145.

[94] Gamberger, D., N. Lavrac, and C. Groselj (1999) "Experiments with noise filtering in a medical domain," in *Proceedings of 16th International Conference on Machine Learning*, Morgan Kaufmann, pp. 143–151.

[95] Gamberger, D., N. Lavrac, and S. Dzeroski (2000) "Noise Detection and Elimination in Data Preprocessing: experiments in medical domains," *Applied Artificial Intelligence*, **14**, pp. 205–223.

[96] Zhu, X., X. Wu, and S. Chen (2003) "Eliminating Class Noise in Large Datasets," in *Proceedings of 20th International Conference on Machine Learning*, pp. 920–927.

[97] Yuille, A. L. and A. Rangarajan (2002) "The Concave-Convex Procedure (CCCP)," in *Advances in Neural Information Processing Systems 14* (T. G. Dietterich, S. Becker, and Z. Ghahramani, eds.), MIT Press, Cambridge, MA.

[98] Smola, A., S. Vishwanathan, and T. Hofmann (2005) "Kernel methods for missing variables," in *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, pp. 325–332.

[99] Shen, X., G. C. Tseng, X. Zhang, and W. H. Wong (2003) "On psi-Learning," *Journal of the American Statistical Association*, **98**, pp. 724–734.

Vita

Şeyda Ertekin

Şeyda Ertekin is a computer scientist and an engineer who likes to work on interesting computational problems. Şeyda is a graduate of Gazi Anadolu Lisesi in Ankara for her mid- and high school education. She received her B.Sc. degree from the Electrical & Electronics Engineering Department in Orta Dogu Teknik Üniversitesi - ODTÜ (Middle East Technical University - METU) in Ankara, Turkey, where she also continued her M.Sc. studies in the fields of electromagnetics and telecommunication. After graduating from ODTÜ, she worked as an Electronics System Engineer at Aselsan Inc. and contributed to several projects on designing and building wide area coverage digital radio telecommunication systems and the integration of various voice and data devices to wireless communication systems. Prior to joining Penn State for her Ph.D. studies in Computer Science and Engineering, she received an M.Sc. degree in Computer Science with a particular focus on data mining from University of Louisiana - Lafayette in USA.

Her Ph.D. studies focused on the design, analysis and implementation of machine learning algorithms and approaches for large scale datasets to solve real-world problems in the fields of data mining, knowledge discovery, information retrieval and artificial intelligence. She is mainly known for her research that spans online and active learning for efficient and scalable Support Vector Machine learning. Throughout her Ph.D. studies, Şeyda also worked as a researcher in the Machine Learning Group at NEC Research Laboratories in Princeton, NJ. At Penn State, part of her work was funded by NEC Labs and she regularly visited and collaborated with researchers at NEC Labs throughout her Ph.D. studies. Şeyda also held a data consultant position at Traffic.com Inc., for a project on providing personalized real-time traffic information. Her papers are published and under review in several top-tier journals and conferences in the fields of machine learning, data mining, information retrieval and knowledge management.

Şeyda has served as the vice-president of Engineering Graduate Student Council (EGSC) of Penn State. She is one of the co-founders and co-chairs of the College of Engineering Research Symposium (CERS), which has since been an annual event at Penn State. Her dissertation got the distinction of being the nominee of CSE department for the PSU Alumni Dissertation Award, which is the most prestigious award at Penn State for graduating Ph.D. students. She is also a member of ACM, IEEE and AAAI.

Şeyda was born to Ayşe and Sadullah Ertekin in a winter morning in Ankara, Turkey. She is married to Levent Bolelli, who is also a doctor of computer science. She is a proud mother of one constant source of amazement and inspiration, their daughter, Alya Su Bolelli. They currently reside in the United States.